

ARMY RESEARCH LABORATORY



Smartphone Application Enabling Global Graph Exploitation and Research

by Mark R. Mittrick

ARL-TR-6439

May 2013

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5066

ARL-TR-6439

May 2013

Smartphone Application Enabling Global Graph Exploitation and Research

Mark R. Mittrick

Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>				
1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE			3. DATES COVERED (From - To)
May 2013				October 2011–August 2012
4. TITLE AND SUBTITLE		Smartphone Application Enabling Global Graph Exploitation and Research		
		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER		
Mark R. Mittrick				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER		
U.S. Army Research Laboratory ATTN: RDRL-CII-C Aberdeen Proving Ground, MD 21005-5066		ARL-TR-6439		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)		
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT				
Approved for public release; distribution is unlimited.				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT				
A new weapon being developed is changing combat as we know it—the smartphone. Smartphone applications have revolutionized information sharing around the globe for military and civilians alike. The U.S. Army Research Laboratory (ARL) is developing a smartphone application that will promote rapid sharing of tactical information between Soldiers in the field and military intelligence analysts in a company intelligence support team. This application allows a Soldier to enter tactical information into the Distributed Common Ground System (DCGS)-A Global Graph to enable near-real-time analysis using the ARL heterogeneous data proximity tool.				
15. SUBJECT TERMS				
smartphone, HDPT, global graph, ozone widget framework, distributed common ground system, web service				
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT			c. THIS PAGE
Unclassified	Unclassified	Unclassified	UU	66
19b. TELEPHONE NUMBER (Include area code) 410-278-4148				

Contents

List of Figures	iv
List of Tables	iv
1. Introduction	1
2. HDPT Report Application	2
2.1 Create Person.....	2
2.2 Modify Person	3
2.3 Settings	4
3. Heterogeneous Data Proximity Tool	5
4. JavaScript Object Notation (JSON) Code Example: REST Web Service	7
5. Sequence of Events	9
6. Conclusion	10
7. References/Other Documentation	10
Appendix. – JAVA/XML Code	11
List of Symbols, Abbreviations, and Acronyms	59
Distribution List	60

List of Figures

Figure 1. HDPT report application: Create Person/example drop down box.....	2
Figure 2. HDPT report application: Modify Person/example drop down box.	4
Figure 3. HDPT report application: settings.....	5
Figure 4. HDPT OWF map interface.....	6
Figure 5. HDPT similarity results.....	6
Figure 6. HDPT report application in the field.....	7
Figure 7. HDPT report application sequence.....	9

List of Tables

Table 1. Data elements and possible values.....	3
-------------------------------------------------	---

1. Introduction

A new weapon being developed is changing combat as we know it—the smartphone. Smartphone applications have revolutionized information sharing around the globe for military and civilians alike. The U.S. Army Research Laboratory (ARL) is developing a smartphone application that will promote rapid sharing of tactical information between Soldiers in the field and military intelligence analysts in a company intelligence support team (COIST).

This application allows a Soldier to enter tactical information into the Distributed Common Ground System (DCGS)-A Global Graph to enable near-real-time analysis using the ARL heterogeneous data proximity tool (HDPT).

This report will focus on ARL's HDPT smartphone application and its interaction with the DCGS-A Global Graph and HDPT at the 2012 On the Move (OTM) Exercise. The OTM exercise, which focuses on cutting-edge technologies for the future force, is held annually at Fort Dix, NJ, and for ARL represented the culmination of several months of work to integrate new technologies into tactical networks.

2. HDPT Report Application

The HDPT report application has three main tabs: “Create Person,” “Modify Person,” and “Settings.” These tabs allow the Soldier to collect intelligence information from the field and perform various tasks with the collected information, such as inputting new information, updating old information, or removing inaccurate information.

2.1 Create Person

When the HDPT report application is initiated, the user is presented with the “Create Person” tab (see figure 1), which allows a Soldier to create a new person of interest. The Solider then enters all of the information from the field into the text and drop-down boxes and submits it via the “Send Report” button to the DCGS-A Global Graph for analysis by HDPT.



Figure 1. HDPT report application: Create Person/example drop down box.

Table 1 shows the possible questions and values when creating a new person for use in this exercise. Not all questions need to be answered for the information to be submitted for analysis.

Table 1. Data elements and possible values.

Data Element Name	Possible Values
Name	Suspect's name
Tribal affiliation	Pashtu, Baloch, Hazara, Tajik
Marital status	Married, Single
Nationality	Muslim, Afghan
Place of birth	Born in area, born outside area
Equipment	Pistol, knife, gang colors, bomb, video camera, cell phone, uniform, briefcase
Vehicle	White panel truck, blue motorcycle, silver compact car, blue minivan, grey sedan, brown pickup truck, black suv, burgundy luxury sedan
Criminal record	Has one, doesn't have one
Education level	High, low
Military record	Not employed, blue collar, white collar
Religion	Mild theology, radical theology
Skill	Photography, writing, electrical, mechanical, computer, driving, financial
Address	Times Square Village, Viet Nam Village, Vertol Village, Hanover Village, Cook Corner Village, Gredge Town, Utes Village, Horizon Village

2.2 Modify Person

The “Modify Person” tab is very similar to the “Create Person” tab. When new information becomes available in the field, the Soldier might need to update the Global Graph. To do so, the Solider must first locate the suspect in question from a list of suspects in the Global Graph (see figure 2). The Soldier can then modify the suspects’ characteristics, as described in section 2.1.

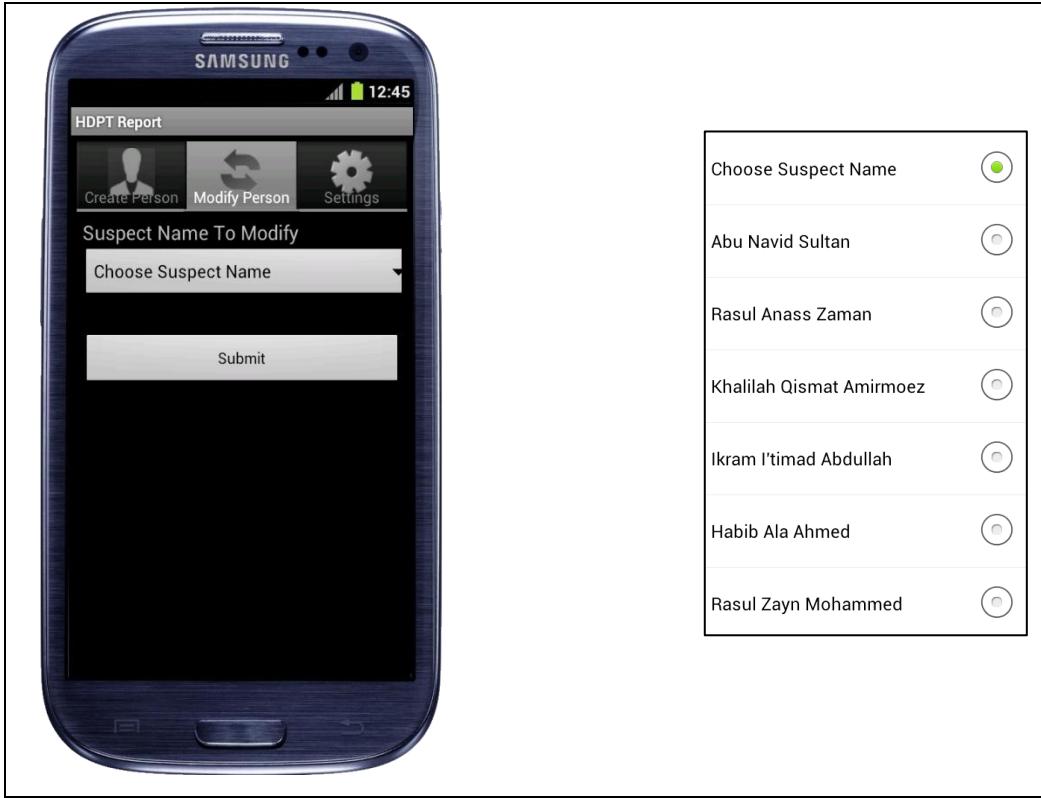


Figure 2. HDPT report application: Modify Person/example drop down box.

2.3 Settings

The “Settings” tab is used to select a DCGS-A Global Graph server (see figure 3). This allows the Soldier the ability to quickly and easily change servers as necessary. In this exercise, we had one operational server and a backup in case the main server was not functioning properly, ensuring that the smartphone application would always be available.



Figure 3. HDPT report application:
settings.

3. Heterogeneous Data Proximity Tool

The HDPT web application built by the ARL is deployable in the ozone widget framework (OWF) and utilizes the Global Graph via web service calls as its data source. HDPT calculates the similarity of individuals, defined by a set attributes, resident within a given population and visualizes the results in a three-dimensional (3-D) scatter plot. To calculate similarity, it uses Gower similarity and multidimensional scaling algorithms contained in the R statistical computing environment. The goal of the tool is to give a military analyst further understanding of the local operational human environment.

Figure 4 shows the HDPT OWF map interface and figure 5 shows the generated results.

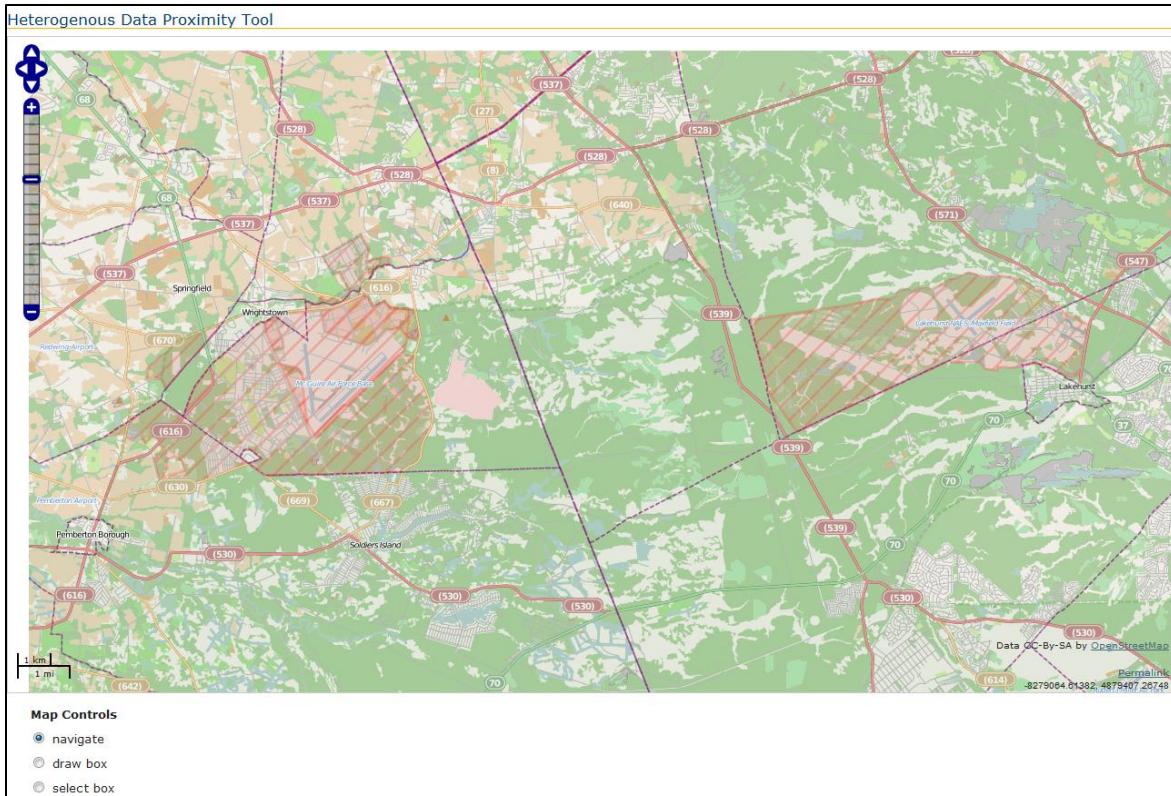


Figure 4. HDPT OWF map interface.

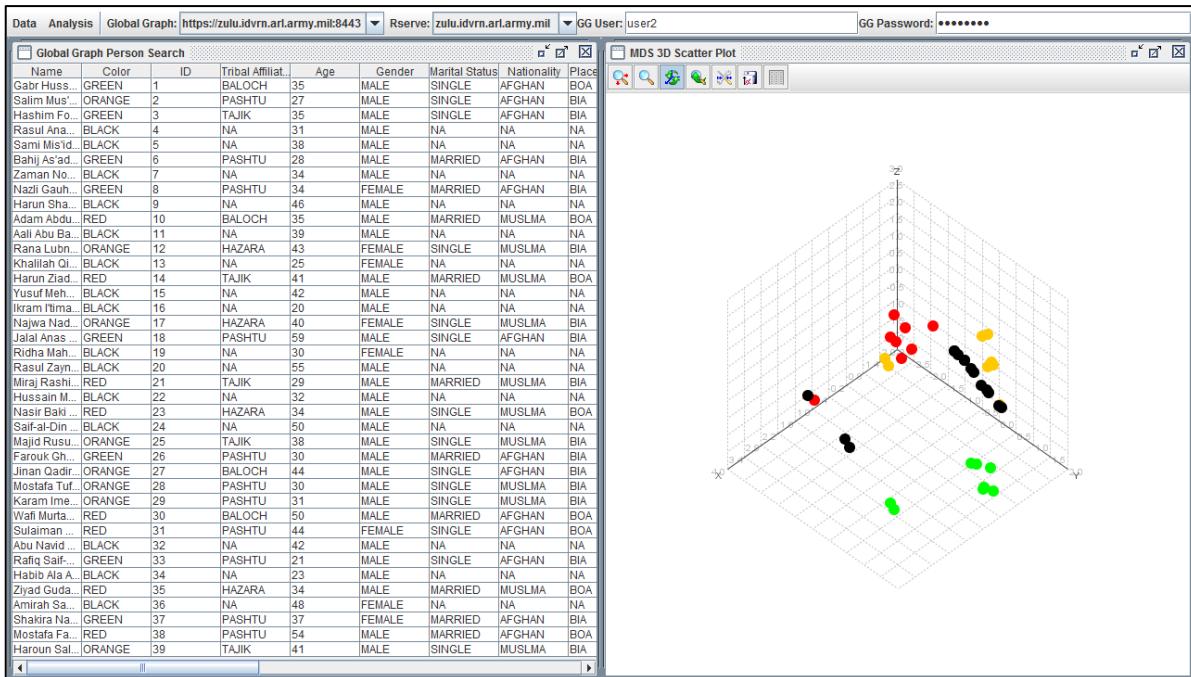


Figure 5. HDPT similarity results.

The HDPT smartphone report application is used in conjunction with HDPT to input new data or update previous data in the Global Graph (see figure 6). The new/updated data is integrated with the previously known data and analyzed by HDPT. The resulting graph is then displayed to the analyst, who can visually compare and contrast the similarity results before making a decision.



Figure 6. HDPT report application in the field.

4. JavaScript Object Notation (JSON) Code Example: REST Web Service

In order to create, update, or search records in the Global Graph, the smartphone application needed to utilize the available representational state transfer (REST) web services. Most of the application development focused on this aspect. These REST web services calls were accomplished by sending a “POST” or “GET” request (depending on what task you are trying to accomplish), with your data encoded in the JavaScript object notation (JSON) format, to the URL <http://localhost/gg/entity/Person>.

The following is an example JSON structure for creating a new person called “Test User” who is of Serbian ethnicity, male gender, 6 ft tall (1.8 m), 180 lb, and the date of birth represented by using milliseconds from epoch. Although this example looks trivial, it usually becomes a lot more complicated as more attributes are added. This is usually due to having nested arrays, having to convert one system to another, or some incomplete information. In the example, note

that height had to be converted from feet to meters and date of birth had to be converted from month/date/year into seconds since epoch. If even one aspect of the JSON structure is not correct, the whole thing will fail.

```
{  
    "entities": {  
        "Person": [ {  
            "names": [ {  
                "fullName": "Test User"  
            } ],  
            "ethnicity": "Serb",  
            "gender": {  
                "physicalValue": "MALE"  
            },  
            "height": {  
                "amount": 1.8  
            },  
            "weight": {  
                "amount": 180  
            },  
            "dateOfBirth": "622735349826"  
        } ]  
    }  
}
```

5. Sequence of Events

Figure 7 illustrates the overall sequence of events for the smartphone application.

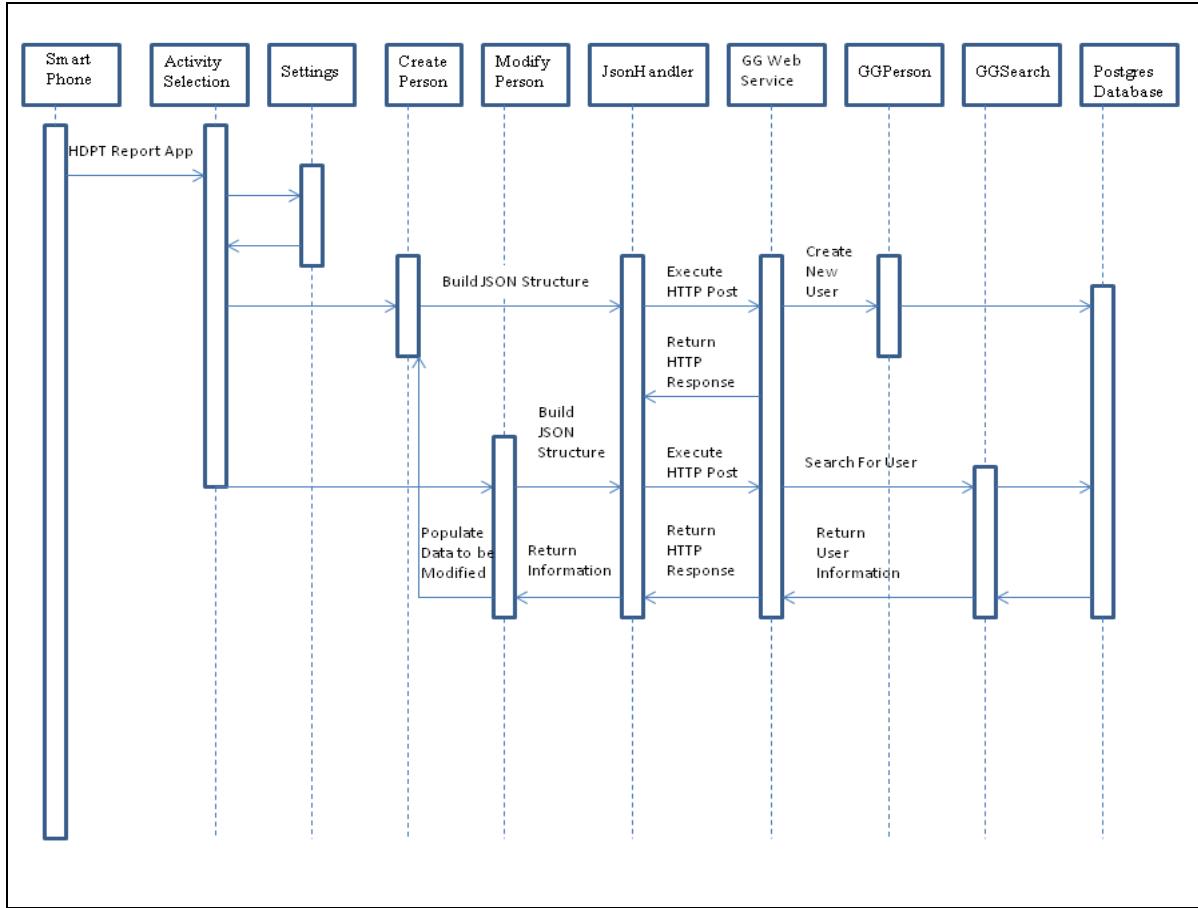


Figure 7. HDPT report application sequence.

6. Conclusion

ARL's HDPT smartphone application allows Soldiers to enter tactical information from the field into the DCGS-A Global Graph for near-real-time analysis. This is the first step in promoting the rapid sharing of information between Soldiers and military intelligence analysts. In the future, additional features and refinements, including value of information, will be added to further enhance ARL's smartphone technology.

7. References/Other Documentation

For additional information please see:

- Mittrick, M.; Richardson, J. *Lessons Learned with a Global Graph and Ozone Widget Framework (OWF) Testbed*; ARL-TR-6440; U.S. Army Research Laboratory: Aberdeen, MD, May 2013.
- Hanratty, T.; Richardson, J. The Heterogeneous Data-Reduction Proximity Tool: A Visual Analytic for High-Dimensional Data Exploitation; U.S. Army Research Laboratory: Aberdeen, MD, in press.
- Potomac Fusion Documentation:
<http://widget.potomacfusion.com/main/home>
<http://www.forge.mil/Community.html>

Appendix. – JAVA/XML Code

This appendix appears in its original form, without editorial change.

EasySSLSocketFactory.java

```
1: package mil.army.arl.hdpt;
2:
3: /*
4: * Licensed to the Apache Software Foundation (ASF) under one
5: * or more contributor license agreements. See the NOTICE file
6: * distributed with this work for additional information
7: * regarding copyright ownership. The ASF licenses this file
8: * to you under the Apache License, Version 2.0 (the
9: * "License"); you may not use this file except in compliance
10: * with the License. You may obtain a copy of the License at
11: *
12: * http://www.apache.org/licenses/LICENSE-2.0
13: *
14: * Unless required by applicable law or agreed to in writing,
15: * software distributed under the License is distributed on an
16: * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
17: * KIND, either express or implied. See the License for the
18: * specific language governing permissions and limitations
19: * under the License.
20: */
21:
22: import java.io.IOException;
23: import java.net.InetAddress;
24: import java.net.InetSocketAddress;
25: import java.net.Socket;
26: import java.net.UnknownHostException;
27:
28: import javax.net.ssl.SSLContext;
29: import javax.net.ssl.SSLSocket;
30: import javax.net.ssl.TrustManager;
31:
32: import org.apache.http.conn.ConnectTimeoutException;
33: import org.apache.http.conn.scheme.LayeredSocketFactory;
34: import org.apache.http.params.HttpConnectionParams;
35: import org.apache.http.params.HttpParams;
36:
37: /**
38: * This socket factory will create ssl socket that accepts self signed
39: * certificate
40: *
41: * @author olamy
42: * @version $Id: EasySSLSocketFactory.java 765355 2009-04-15 20:59:07Z evenisse
43: * $S
44: * @since 1.2.3
45: */
46: public class EasySSLSocketFactory implements LayeredSocketFactory {
47:
48: private SSLContext sslcontext = null;
49:
50: private static SSLContext createEasySSLContext() throws IOException {
51: try {
52: SSLContext context = SSLContext.getInstance("TLS");
53: context.init(null, new TrustManager[] { new EasyX509TrustManager(
54: null) }, null);
55: return context;
56: } catch (Exception e) {
57: throw new IOException(e.getMessage());
58: }
59: }
60:
61: private SSLContext getSSLContext() throws IOException {
62: if (this.sslcontext == null) {
63: this.sslcontext = createEasySSLContext();
64: }
65: return this.sslcontext;
66: }
67:
68: /**
69: * @see org.apache.http.conn.scheme.SocketFactory#connectSocket(java.net.Socket,
70: * java.lang.String, int, java.net.InetAddress, int,
71: * org.apache.http.params.HttpParams)
72: */
73: public Socket connectSocket(Socket sock, String host, int port,
74: InetAddress localAddress, int localPort, HttpParams params)
75: throws IOException, UnknownHostException, ConnectTimeoutException {
```

EasySSLocketFactory.java

```
76: int connTimeout = HttpConnectionParams.getConnectionTimeout(params);
77: int soTimeout = HttpConnectionParams.getSoTimeout(params);
78:
79: InetSocketAddress remoteAddress = new InetSocketAddress(host, port);
80: SSLSocket sslsock = (SSLSocket) ((sock != null) ? sock : createSocket());
81:
82: if ((localAddress != null) || (localPort > 0)) {
83: // we need to bind explicitly
84: if (localPort < 0) {
85: localPort = 0; // indicates "any"
86: }
87: InetSocketAddress isa = new InetSocketAddress(localAddress,
88: localPort);
89: sslsock.bind(isa);
90: }
91:
92: sslsock.connect(remoteAddress, connTimeout);
93: sslsock.setSoTimeout(soTimeout);
94: return sslsock;
95:
96: }
97:
98: /**
99: * @see org.apache.http.conn.scheme.SocketFactory#createSocket()
100: */
101: //public Socket createSocket() throws IOException {
102: //return getSSLContext().getSocketFactory().createSocket();
103: //}
104:
105: /**
106: * @see org.apache.http.conn.scheme.SocketFactory#isSecure(java.net.Socket)
107: */
108: public boolean isSecure(Socket socket) throws IllegalArgumentException {
109: return true;
110: }
111:
112: /**
113: * @see org.apache.http.conn.scheme.LayeredSocketFactory#createSocket(java.net.Socket,
114: * java.lang.String, int, boolean)
115: */
116: //public Socket createSocket(Socket socket, String host, int port,
117: // boolean autoClose) throws IOException, UnknownHostException {
118: // return getSSLContext().getSocketFactory().createSocket(socket, host, port, autoClose);
119: //}
120:
121: // -----
122: // javadoc in org.apache.http.conn.scheme.SocketFactory says :
123: // Both Object.equals() and Object.hashCode() must be overridden
124: // for the correct operation of some connection managers
125: // -----
126:
127: public boolean equals(Object obj) {
128: return ((obj != null) && obj.getClass().equals(
129: EasySSLocketFactory.class));
130: }
131:
132: public int hashCode() {
133: return EasySSLocketFactory.class.hashCode();
134: }
135:
136: @Override
137: public Socket createSocket(Socket socket, String host, int port, boolean autoClose) throws IOException, UnknownHostException {
138: // TODO Auto-generated method stub
139: return getSSLContext().getSocketFactory().createSocket(socket, host, port, autoClose);
140: }
141:
142: @Override
143: public Socket createSocket() throws IOException {
144: // TODO Auto-generated method stub
145: return null;
146: }
147:
148: }
```

EasyX509TrustManager.java

```
1: package mil.army.arl.hdpt;
2:
3: /*
4: * Licensed to the Apache Software Foundation (ASF) under one
5: * or more contributor license agreements. See the NOTICE file
6: * distributed with this work for additional information
7: * regarding copyright ownership. The ASF licenses this file
8: * to you under the Apache License, Version 2.0 (the
9: * "License"); you may not use this file except in compliance
10: * with the License. You may obtain a copy of the License at
```

```

11: *
12: * http://www.apache.org/licenses/LICENSE-2.0
13: *
14: * Unless required by applicable law or agreed to in writing,
15: * software distributed under the License is distributed on an
16: * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
17: * KIND, either express or implied. See the License for the
18: * specific language governing permissions and limitations
19: * under the License.
20: */
21:
22: import java.security.KeyStore;
23: import java.security.KeyStoreException;
24: import java.security.NoSuchAlgorithmException;
25: import java.security.cert.CertificateException;
26: import java.security.cert.X509Certificate;
27:
28: import javax.net.ssl.TrustManager;
29: import javax.net.ssl.TrustManagerFactory;
30: import javax.net.ssl.X509TrustManager;
31:
32: /**
33: * @author olamy
34: * @version $Id: EasyX509TrustManager.java 765355 2009-04-15 20:59:07Z evenisse $
35: * @since 1.2.3
36: */
37: public class EasyX509TrustManager
38: implements X509TrustManager
39: {
40:
41: private X509TrustManager standardTrustManager = null;
42:
43: /**
44: * Constructor for EasyX509TrustManager.
45: */
46: public EasyX509TrustManager( KeyStore keystore )
47: throws NoSuchAlgorithmException, KeyStoreException
48: {
49: super();
50: TrustManagerFactory factory = TrustManagerFactory.getInstance( TrustManagerFactory.getDefaultAlgorithm() );
51: factory.init( keystore );
52: TrustManager[] trustmanagers = factory.getTrustManagers();
53: if ( trustmanagers.length == 0 )
54: {
55: throw new NoSuchAlgorithmException( "no trust manager found" );
56: }
57: this.standardTrustManager = (X509TrustManager) trustmanagers[0];
58: }
59:
60: /**
61: * @see javax.net.ssl.X509TrustManager#checkClientTrusted(X509Certificate[],String authType)
62: */
63: public void checkClientTrusted( X509Certificate[] certificates, String authType )
64: throws CertificateException
65: {
66: standardTrustManager.checkClientTrusted( certificates, authType );
67: }
68:
69: /**
70: * @see javax.net.ssl.X509TrustManager#checkServerTrusted(X509Certificate[],String authType)
71: */
72: public void checkServerTrusted( X509Certificate[] certificates, String authType )
73: throws CertificateException
74: {
75: if ( ( certificates != null ) && ( certificates.length == 1 ) )

```

EasyX509TrustManager.java

```

76: {
77: certificates[0].checkValidity();
78: }
79: else
80: {
81: // standardTrustManager.checkServerTrusted( certificates, authType );
82: }
83: }
84:
85: /**
86: * @see javax.net.ssl.X509TrustManager#getAcceptedIssuers()
87: */
88: public X509Certificate[] getAcceptedIssuers()
89: {
90: return this.standardTrustManager.getAcceptedIssuers();

```

```

91: }
92: }
93: }

hdpt.java
1: package mil.army.arl.hdpt;
2:
3: import mil.army.arl.hdpt.R;
4: import android.app.Activity;
5: import android.content.Intent;
6: import android.os.Bundle;
7: import android.os.Handler;
8:
9: public class hdpt extends Activity {
10:
11: private final int SPLASH_DISPLAY_LENGTH = 3000;
12:
13: /** Called when the activity is first created. */
14:
15: @Override
16:
17: public void onCreate(Bundle icicle) {
18: super.onCreate(icicle);
19: setContentView(R.layout.splash);
20: /* New Handler to start the Menu-Activity
21: * and close this Splash-Screen after some seconds.*/
22: new Handler().postDelayed(new Runnable(){
23: @Override
24: public void run() {
25: /* Create an Intent that will start the Menu-Activity. */
26: Intent mainIntent = new Intent(hdpt.this,main.class);
27: hdpt.this.startActivity(mainIntent);
28: hdpt.this.finish();
29: }
30: }, SPLASH_DISPLAY_LENGTH);
31: }
32:
33: }//end

```

```

main.java
1: package mil.army.arl.hdpt;
2:
3: import android.app.TabActivity;
4: import android.content.Context;
5: import android.content.Intent;
6: import android.content.res.Resources;
7: import android.net.ConnectivityManager;
8: import android.os.Bundle;
9: import android.view.Gravity;
10: import android.widget.TabHost;
11: import android.widget.Toast;
12:
13: public class main extends TabActivity {
14:
15: boolean reset=main3.reset;
16:
17:
18: public boolean isOnline() {
19: ConnectivityManager cm =
20: (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
21:
22: return cm.getActiveNetworkInfo() != null &&
23: cm.getActiveNetworkInfo().isConnectedOrConnecting();
24: }
25:
26:
27: public void onCreate(Bundle savedInstanceState) {
28: super.onCreate(savedInstanceState);
29:
30:
31: if (isOnline()){
32: //System.out.println("Online");
33: //Toast msg = Toast.makeText(this, "Internet connection available.", Toast.LENGTH_LONG);
34: //msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
35: //msg.show();
36: }
37: else {
38: //System.out.println("Offline");
39: Toast msg = Toast.makeText(this, "No Internet connection available.\n Please try again later.", Toast.LENGTH_LONG);
40: msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
41: msg.show();
42: this.finish();
43: }
44:
45: setContentView(R.layout.main);
46:
47: Resources res = getResources(); // Resource object to get Drawables
48: TabHost tabHost = getTabHost(); // The activity TabHost

```

```

49: TabHost.TabSpec spec; // Resusable TabSpec for each tab
50: Intent intent; // Reusable Intent for each tab
51:
52: // Create an Intent to launch an Activity for the tab (to be reused)
53: intent = new Intent().setClass(this, main2.class);
54:
55: // Initialize a TabSpec for each tab and add it to the TabHost
56: spec = tabHost.newTabSpec("Create Person").setIndicator("Create Person",
res.getDrawable(R.drawable.person_create)).setContent(intent);
57:
58: try{
59: tabHost.addTab(spec);
60: }
61: catch(Exception e) {
62: System.out.println("Error " + e);
63: }
64:
65: // Do the same for the other tabs
66: intent = new Intent().setClass(this, main3.class);
67:
68: // Initialize a TabSpec for each tab and add it to the TabHost
69: spec = tabHost.newTabSpec("Modify Person").setIndicator("Modify Person",
res.getDrawable(R.drawable.person_modify)).setContent(intent);
70:
71: try{
72: tabHost.addTab(spec);
73: }
74: catch(Exception e) {
75: System.out.println("Error " + e);
}

main.java
76: }
77:
78:
79: // Do the same for the other tabs
80: intent = new Intent().setClass(this, settings.class);
81:
82: // Initialize a TabSpec for each tab and add it to the TabHost
83: spec = tabHost.newTabSpec("Settings").setIndicator("Settings", res.getDrawable(R.drawable.settings)).setContent(intent);
84:
85: try{
86: tabHost.addTab(spec);
87: }
88: catch(Exception e) {
89: System.out.println("Error " + e);
90: }
91:
92: if (reset==true){
93: tabHost.setCurrentTab(1);
94: }
95:
96: else {
97: tabHost.setCurrentTab(1);
98: }
99: }

100: }

main2.java
1: package mil.army.arl.hdpt;
2:
3:
4: import java.text.ParseException;
5: import java.util.Calendar;
6: import java.util.Date;
7: import android.app.Activity;
8: import android.app.DatePickerDialog;
9: import android.app.Dialog;
10: import android.os.Bundle;
11: import android.view.Gravity;
12: import android.view.View;
13: import android.widget.ArrayAdapter;
14: import android.widget.Button;
15: import android.widget.DatePicker;
16: import android.widget.EditText;
17: import android.widget.Spinner;
18: import android.widget.TextView;
19: import android.widget.Toast;
20: import java.io.UnsupportedEncodingException;
21: import org.apache.http.HttpResponse;
22: import org.apache.http.client.HttpClient;
23: import org.apache.http.client.methods.HttpPost;
24: import org.apache.http.conn.ClientConnectionManager;
25: import org.apache.http.params.ConnManagerPNames;
26: import org.apache.http.params.ConnPerRouteBean;
27: import org.apache.http.conn.scheme.PlainSocketFactory;
28: import org.apache.http.conn.scheme.Scheme;

```

```

29: import org.apache.http.conn.scheme.SchemeRegistry;
30: import org.apache.http.entity.StringEntity;
31: import org.apache.http.impl.client.DefaultHttpClient;
32: import org.apache.http.impl.conn.SingleClientConnManager;
33: import org.apache.http.message.BasicHeader;
34: import org.apache.http.message.BasicHttpRequest;
35: import org.apache.http.params.BasicHttpParams;
36: import org.apache.http.params.HttpParams;
37: import org.apache.http.params.HttpProtocolParams;
38: import org.apache.http.protocol.HTTP;
39: import org.apache.http.Header;
40: import org.apache.http.HttpStatus;
41: import org.apache.http.HttpVersion;
42: import org.apache.http.auth.UsernamePasswordCredentials;
43: import org.apache.http.impl.auth.DigestScheme;
44: import org.codehaus.jackson.map.ObjectMapper;
45: import org.codehaus.jackson.node.ArrayNode;
46: import org.codehaus.jackson.node.ObjectNode;
47:
48: public class main2 extends Activity {
49:
50:     private TextView mDateDisplay;
51:     private Button mPickDate, mSubmit;
52:     private int mYear;
53:     private int mMonth;
54:     private int mDay;
55:     int age;
56:     int gender_pos;
57:     //int height_pos;
58:     //int weight_pos;
59:     long epoch;
60:
61:     public static final double feet2meters = 0.305;
62:     static final int DATE_DIALOG_ID = 0;
63:
64:     // Called when the activity is first created
65:     @Override
66:     public void onCreate(Bundle savedInstanceState) {
67:         super.onCreate(savedInstanceState);
68:         setContentView(R.layout.form);
69:
70:         // capture our View elements
71:         mDateDisplay = (TextView) findViewById(R.id.dateDisplay);
72:         mPickDate = (Button) findViewById(R.id.pickDate);
73:         mSubmit = (Button) findViewById(R.id.ButtonSendFeedback);
74:
75:         // add a click listener to the button

```

main2.java

```

76: mSubmit.setOnClickListener(new View.OnClickListener() {
77:     public void onClick(View x) {
78:         sendFeedback(x);
79:     }
80: });
81:
82: // add a click listener to the button
83: mPickDate.setOnClickListener(new View.OnClickListener() {
84:     public void onClick(View v) {
85:         showDialog(DATE_DIALOG_ID);
86:     }
87: });
88:
89: // get the current date
90: final Calendar c = Calendar.getInstance();
91: mYear = c.get(Calendar.YEAR);
92: mMonth = c.get(Calendar.MONTH);
93: mDay = c.get(Calendar.DAY_OF_MONTH);
94:
95: // display the current date (this method is below)
96: updateDisplay();
97:
98:
99: ArrayAdapter<CharSequence> adapter0 = ArrayAdapter.createFromResource(
100: this, R.array.nationality_array, android.R.layout.simple_spinner_item );
101: adapter0.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
102:
103: Spinner s0 = (Spinner) findViewById( R.id.nationality_spinner );
104: s0.setAdapter( adapter0 );
105:
106: ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
107: this, R.array.gender_array, android.R.layout.simple_spinner_item );
108: adapter1.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
109:
110: Spinner s1 = (Spinner) findViewById( R.id.gender_spinner );
111: s1.setAdapter( adapter1 );
112:
113: /*
114: ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(

```

```

115: this, R.array.height_array, android.R.layout.simple_spinner_item );
116: adapter2.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
117:
118:
119: Spinner s2 = (Spinner) findViewById( R.id.height_spinner );
120: s2.setAdapter( adapter2 );
121: */
122:
123: ArrayAdapter<CharSequence> adapter21 = ArrayAdapter.createFromResource(
124: this, R.array.marital_array, android.R.layout.simple_spinner_item );
125: adapter21.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
126:
127: Spinner s21 = (Spinner) findViewById( R.id.marital_spinner );
128: s21.setAdapter( adapter21 );
129:
130: ArrayAdapter<CharSequence> adapter22 = ArrayAdapter.createFromResource(
131: this, R.array.birthplace_array, android.R.layout.simple_spinner_item );
132: adapter22.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
133:
134: Spinner s22 = (Spinner) findViewById( R.id.birthplace_spinner );
135: s22.setAdapter( adapter22 );
136:
137: /*
138: ArrayAdapter<CharSequence> adapter3 = ArrayAdapter.createFromResource(
139: this, R.array.weight_array, android.R.layout.simple_spinner_item );
140: adapter3.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
141:
142: Spinner s3 = (Spinner) findViewById( R.id.weight_spinner );
143: s3.setAdapter( adapter3 );
144: */
145:
146: ArrayAdapter<CharSequence> adapter4 = ArrayAdapter.createFromResource(
147: this, R.array.equipment_array, android.R.layout.simple_spinner_item );
148: adapter4.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
149:
150: Spinner s4 = (Spinner) findViewById( R.id.equipment_spinner );

main2.java
151: s4.setAdapter( adapter4 );
152:
153:
154: ArrayAdapter<CharSequence> adapter5 = ArrayAdapter.createFromResource(
155: this, R.array.vehicle_array, android.R.layout.simple_spinner_item );
156: adapter5.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
157:
158: Spinner s5 = (Spinner) findViewById( R.id.vehicle_spinner );
159: s5.setAdapter( adapter5 );
160:
161:
162: ArrayAdapter<CharSequence> adapter6 = ArrayAdapter.createFromResource(
163: this, R.array.criminalrecord_array, android.R.layout.simple_spinner_item );
164: adapter6.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
165:
166: Spinner s6 = (Spinner) findViewById( R.id.criminalrecord_spinner );
167: s6.setAdapter( adapter6 );
168:
169:
170: ArrayAdapter<CharSequence> adapter7 = ArrayAdapter.createFromResource(
171: this, R.array.education_array, android.R.layout.simple_spinner_item );
172: adapter7.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
173:
174: Spinner s7 = (Spinner) findViewById( R.id.education_spinner );
175: s7.setAdapter( adapter7 );
176:
177:
178: ArrayAdapter<CharSequence> adapter8 = ArrayAdapter.createFromResource(
179: this, R.array.employment_array, android.R.layout.simple_spinner_item );
180: adapter8.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
181:
182: Spinner s8 = (Spinner) findViewById( R.id.employment_spinner );
183: s8.setAdapter( adapter8 );
184:
185: ArrayAdapter<CharSequence> adapter9 = ArrayAdapter.createFromResource(
186: this, R.array.religion_array, android.R.layout.simple_spinner_item );
187: adapter9.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
188:
189: Spinner s9 = (Spinner) findViewById( R.id.religion_spinner );
190: s9.setAdapter( adapter9 );
191:
192:
193: ArrayAdapter<CharSequence> adapter10 = ArrayAdapter.createFromResource(
194: this, R.array.skill_array, android.R.layout.simple_spinner_item );
195: adapter10.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
196:
197: Spinner s10 = (Spinner) findViewById( R.id.skill_spinner );

```

```

198: s10.setAdapter( adapter10 );
199:
200:
201: ArrayAdapter<CharSequence> adapter11 = ArrayAdapter.createFromResource(
202: this, R.array.address_array, android.R.layout.simple_spinner_item );
203: adapter11.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
204:
205: Spinner s11 = (Spinner) findViewById( R.id.address_spinner );
206: s11.setAdapter( adapter11 );
207:
208:
209: ArrayAdapter<CharSequence> adapter12 = ArrayAdapter.createFromResource(
210: this, R.array.tribe_array, android.R.layout.simple_spinner_item );
211: adapter12.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
212:
213: Spinner s12 = (Spinner) findViewById( R.id.tribe_spinner );
214: s12.setAdapter( adapter12 );
215:
216: } //onCreate
217:
218: // On Resume
219: @Override
220: public void onResume() {
221: super.onResume();
222: setContentView(R.layout.form);
223:
224: // capture our View elements

225: mDateDisplay = (TextView) findViewById(R.id.dateDisplay);

main2.java
226: mPickDate = (Button) findViewById(R.id.pickDate);
227: mSubmit = (Button) findViewById(R.id.ButtonSendFeedback);
228:
229: // add a click listener to the button
230: mSubmit.setOnClickListener(new View.OnClickListener() {
231: public void onClick(View x) {
232: sendFeedback(x);
233: }
234: });
235:
236: // add a click listener to the button
237: mPickDate.setOnClickListener(new View.OnClickListener() {
238: public void onClick(View v) {
239: showDialog(DATE_DIALOG_ID);
240: }
241: });
242:
243: // get the current date
244: final Calendar c = Calendar.getInstance();
245: mYear = c.get(Calendar.YEAR);
246: mMonth = c.get(Calendar.MONTH);
247: mDay = c.get(Calendar.DAY_OF_MONTH);
248:
249: // display the current date (this method is below)
250: updateDisplay();
251:
252:
253: ArrayAdapter<CharSequence> adapter0 = ArrayAdapter.createFromResource(
254: this, R.array.nationality_array, android.R.layout.simple_spinner_item );
255: adapter0.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
256:
257: Spinner s0 = (Spinner) findViewById( R.id.nationality_spinner );
258: s0.setAdapter( adapter0 );
259:
260: ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
261: this, R.array.gender_array, android.R.layout.simple_spinner_item );
262: adapter1.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
263:
264: Spinner s1 = (Spinner) findViewById( R.id.gender_spinner );
265: s1.setAdapter( adapter1 );
266:
267: /*
268: ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(
269: this, R.array.height_array, android.R.layout.simple_spinner_item );
270: adapter2.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
271:
272:
273: Spinner s2 = (Spinner) findViewById( R.id.height_spinner );
274: s2.setAdapter( adapter2 );
275: */
276:
277: ArrayAdapter<CharSequence> adapter21 = ArrayAdapter.createFromResource(
278: this, R.array.marital_array, android.R.layout.simple_spinner_item );
279: adapter21.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
280:
281: Spinner s21 = (Spinner) findViewById( R.id.marital_spinner );
282: s21.setAdapter( adapter21 );
283:
```

```

284: ArrayAdapter<CharSequence> adapter22 = ArrayAdapter.createFromResource(
285:     this, R.array.birthplace_array, android.R.layout.simple_spinner_item );
286: adapter22.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
287:
288: Spinner s22 = (Spinner) findViewById( R.id.birthplace_spinner );
289: s22.setAdapter( adapter22 );
290:
291: /*
292: ArrayAdapter<CharSequence> adapter3 = ArrayAdapter.createFromResource(
293: this, R.array.weight_array, android.R.layout.simple_spinner_item );
294: adapter3.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
295:
296: Spinner s3 = (Spinner) findViewById( R.id.weight_spinner );
297: s3.setAdapter( adapter3 );
298: */
299:

300: ArrayAdapter<CharSequence> adapter4 = ArrayAdapter.createFromResource(
main2.java
301: this, R.array.equipment_array, android.R.layout.simple_spinner_item );
302: adapter4.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
303:
304: Spinner s4 = (Spinner) findViewById( R.id.equipment_spinner );
305: s4.setAdapter( adapter4 );
306:
307:
308: ArrayAdapter<CharSequence> adapter5 = ArrayAdapter.createFromResource(
309: this, R.array.vehicle_array, android.R.layout.simple_spinner_item );
310: adapter5.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
311:
312: Spinner s5 = (Spinner) findViewById( R.id.vehicle_spinner );
313: s5.setAdapter( adapter5 );
314:
315:
316: ArrayAdapter<CharSequence> adapter6 = ArrayAdapter.createFromResource(
317: this, R.array.criminalrecord_array, android.R.layout.simple_spinner_item );
318: adapter6.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
319:
320: Spinner s6 = (Spinner) findViewById( R.id.criminalrecord_spinner );
321: s6.setAdapter( adapter6 );
322:
323:
324: ArrayAdapter<CharSequence> adapter7 = ArrayAdapter.createFromResource(
325: this, R.array.education_array, android.R.layout.simple_spinner_item );
326: adapter7.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
327:
328: Spinner s7 = (Spinner) findViewById( R.id.education_spinner );
329: s7.setAdapter( adapter7 );
330:
331:
332: ArrayAdapter<CharSequence> adapter8 = ArrayAdapter.createFromResource(
333: this, R.array.employment_array, android.R.layout.simple_spinner_item );
334: adapter8.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
335:
336: Spinner s8 = (Spinner) findViewById( R.id.employment_spinner );
337: s8.setAdapter( adapter8 );
338:
339: ArrayAdapter<CharSequence> adapter9 = ArrayAdapter.createFromResource(
340: this, R.array.religion_array, android.R.layout.simple_spinner_item );
341: adapter9.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
342:
343: Spinner s9 = (Spinner) findViewById( R.id.religion_spinner );
344: s9.setAdapter( adapter9 );
345:
346:
347: ArrayAdapter<CharSequence> adapter10 = ArrayAdapter.createFromResource(
348: this, R.array.skill_array, android.R.layout.simple_spinner_item );
349: adapter10.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
350:
351: Spinner s10 = (Spinner) findViewById( R.id.skill_spinner );
352: s10.setAdapter( adapter10 );
353:
354:
355: ArrayAdapter<CharSequence> adapter11 = ArrayAdapter.createFromResource(
356: this, R.array.address_array, android.R.layout.simple_spinner_item );
357: adapter11.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
358:
359: Spinner s11 = (Spinner) findViewById( R.id.address_spinner );
360: s11.setAdapter( adapter11 );
361:
362:
363: ArrayAdapter<CharSequence> adapter12 = ArrayAdapter.createFromResource(
364: this, R.array.tribe_array, android.R.layout.simple_spinner_item );
365: adapter12.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
366:
367: Spinner s12 = (Spinner) findViewById( R.id.tribe_spinner );
368: s12.setAdapter( adapter12 );
369:
```

```

370:
371: } //onResume
372:
373: // updates the date in the TextView
374: private void updateDisplay() {

375: mDateDisplay.setText(

main2.java
376: new StringBuilder()
377: // Month is 0 based so add 1
378: .append(mMonth + 1).append("-")
379: .append(mDay).append("-")
380: .append(mYear).append(" "));
381: try {
382: String nMonth = String.format("%02d", (mMonth+1));
383: String nDay = String.format("%02d", mDay);
384: String eDate = (nMonth + "/" + nDay + "/" + mYear);
385:
386: epoch = new java.text.SimpleDateFormat("MM/dd/yyyy").parse(eDate).getTime();
387:
388:
389: Date now = new Date();
390: int month = mMonth;
391: int day = mDay;
392: int year = mYear;
393:
394: int nowMonth = now.getMonth() + 1;
395: int nowYear = now.getYear() + 1900;
396: int result = nowYear - year;
397:
398: if (month > nowMonth) {
399: result--;
400: }
401: else if (month == nowMonth) {
402: int nowDay = now.getDate();
403:
404: if (day > nowDay) {
405: result--;
406: }
407: }
408: age = result;
409:
410: } catch (ParseException e) {
411: e.printStackTrace();
412: }
413: } //updateDisplay
414:
415: // the callback received when the user "sets" the date in the dialog
416: private DatePickerDialog.OnDateSetListener mDateSetListener =
417: new DatePickerDialog.OnDateSetListener() {
418:
419: public void onDateSet(DatePicker view, int year,
420: int monthOfYear, int dayOfMonth) {
421: mYear = year;
422: mMonth = monthOfYear;
423: mDay = dayOfMonth;
424: updateDisplay();
425: }
426: };
427:
428: @Override
429: protected Dialog onCreateDialog(int id) {
430: switch (id) {
431: case DATE_DIALOG_ID:
432: return new DatePickerDialog(this,
433: mDateSetListener,
434: mYear, mMonth, mDay);
435: }
436: return null;
437: }
438:
439: public void sendFeedback(View button) {
440:
441: final EditText nameField = (EditText) findViewById(R.id.EditTextName);
442: String name = nameField.getText().toString();
443:
444: Spinner s = (Spinner) findViewById( R.id.nationality_spinner );
445: String nationality = s.getSelectedItem().toString();
446:
447: Spinner s2 = (Spinner) findViewById( R.id.gender_spinner );
448: String gender = s2.getSelectedItem().toString();
449:
450: Spinner s21 = (Spinner) findViewById( R.id.marital_spinner );

```

```

main2.java
451: String marital = s21.getSelectedItem().toString();
452:
453: Spinner s22 = (Spinner) findViewById( R.id.birthplace_spinner );
454: String birthplace = s22.getSelectedItem().toString();
455:
456: /*
457: Spinner s3 = (Spinner) findViewById( R.id.height_spinner );
458: String height = s3.getSelectedItem().toString();
459:
460: Spinner s4 = (Spinner) findViewById( R.id.weight_spinner );
461: String weight = s4.getSelectedItem().toString();
462: */
463:
464: Spinner s4 = (Spinner) findViewById( R.id.equipment_spinner );
465: String equipment = s4.getSelectedItem().toString();
466:
467: Spinner s5 = (Spinner) findViewById( R.id.vehicle_spinner );
468: String vehicle = s5.getSelectedItem().toString();
469:
470: Spinner s6 = (Spinner) findViewById( R.id.criminalrecord_spinner );
471: String criminalrecord = s6.getSelectedItem().toString();
472:
473: Spinner s7 = (Spinner) findViewById( R.id.education_spinner );
474: String education = s7.getSelectedItem().toString();
475:
476: Spinner s8 = (Spinner) findViewById( R.id.employment_spinner );
477: String employment = s8.getSelectedItem().toString();
478:
479: Spinner s9 = (Spinner) findViewById( R.id.religion_spinner );
480: String religion = s9.getSelectedItem().toString();
481:
482: Spinner s10 = (Spinner) findViewById( R.id.skill_spinner );
483: String skill = s10.getSelectedItem().toString();
484:
485: Spinner s11 = (Spinner) findViewById( R.id.address_spinner );
486: String address = s11.getSelectedItem().toString();
487:
488: Spinner s12 = (Spinner) findViewById( R.id.tribe_spinner );
489: String tribe = s12.getSelectedItem().toString();
490:
491: final TextView dateField = (TextView) findViewById(R.id.dateDisplay);
492: String date = dateField.getText().toString();
493:
494: if(false)
495: {
496: //do nothing
497: }
498:
499: /*
500: if(name.isEmpty() || nationality.equals("Choose Suspect Nationality") || gender.equals("Choose Suspect Gender") ||
marital.equals("Choose Suspect Marital Status")
501: || birthplace.equals("Choose Suspect Place of Birth") || equipment.equals("Choose Suspect Equipment") ||
vehicle.equals("Choose Suspect Vehicle")
502: || criminalrecord.equals("Choose Suspect Criminal Record") || education.equals("Choose Suspect Education") ||
employment.equals("Choose Suspect Employment")
503: || religion.equals("Choose Suspect Religion") || skill.equals("Choose Suspect Skill") || address.equals("Choose Suspect
Address"))
504: {
505: Toast msg = Toast.makeText(this, "Please enter information for all fields!", Toast.LENGTH_LONG);
506: msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
507: msg.show();
508: }
509: */
510:
511: else {
512: //double oHeight = Double.parseDouble( height );
513: //double mHeight = convert(oHeight);
514: //String nHeight = Double.toString(mHeight);
515: double lat = 0, lon = 0;
516:
517: if (nationality.equals("Arab")){
518: nationality = "Muslma";
519: }
520:
521: if (birthplace.equals("Born in Area")){
522: birthplace = "BIA";
523: }
524:
525: if (birthplace.equals("Born outside Area")){

```

```

main2.java
526: birthplace = "BOA";
527: }
528:
529: if (criminalrecord.equals("Has One")) {
530: criminalrecord = "Guilty";
531: }
532:
533: if (criminalrecord.equals("Does Not Have One")) {
534: criminalrecord = "None";
535: }
536:
537: if (employment.equals("Not Employed")) {
538: employment = "NE";
539: }
540:
541: if (employment.equals("White Collar")) {
542: employment = "WC";
543: }
544:
545: if (employment.equals("Blue Collar")) {
546: employment = "BC";
547: }
548:
549: if (religion.equals("Mild Theology")) {
550: religion = "Mld";
551: }
552:
553: if (religion.equals("Radical Theology")) {
554: religion = "Rad";
555: }
556:
557: if (skill.equals("Photography")) {
558: skill = "PH";
559: }
560:
561: if (skill.equals("Writing")) {
562: skill = "WR";
563: }
564:
565: if (skill.equals("Electrical")) {
566: skill = "EL";
567: }
568:
569: if (skill.equals("Mechanical")) {
570: skill = "ME";
571: }
572:
573: if (skill.equals("Computer")) {
574: skill = "CO";
575: }
576:
577: if (skill.equals("Driving")) {
578: skill = "DR";
579: }
580:
581: if (skill.equals("Financial")) {
582: skill = "FI";
583: }
584:
585: if (address.equals("Times Square Village")) {
586: address = "TSV";
587: lat = 39.983611;
588: lon = -74.43;
589: }
590:
591: if (address.equals("Viet Nam Village")) {
592: address = "VNV";
593: lat = 39.97883;
594: lon = -74.43;
595: }
596:
597: if (address.equals("Vertol Village")) {
598: address = "VV";
599: lat = 39.97278;
600: lon = -74.4275;

```

```

main2.java
601: }
602:
603: if (address.equals("Hanover Village")) {
604: address = "HAV";
605: lat = 40.00944;
606: lon = -74.5575;
607: }
608:
609: if (address.equals("Cook Corner Village")) {

```

```

610: address = "CCV";
611: lat = 40.01556;
612: lon = -74.5525;
613: }
614:
615: if (address.equals("Gredge Town")){
616: address = "GT";
617: lat = 40.03083;
618: lon = -74.52444;
619: }
620:
621: if (address.equals("Utes Village")){
622: address = "UV";
623: lat = 40.04583;
624: lon = -74.46056;
625: }
626:
627: if (address.equals("Horizon Village")){
628: address = "HGV";
629: lat = 40.00667;
630: lon = -74.45111;
631: }
632:
633: TextView resultArea;
634: resultArea = new TextView(this);
635: resultArea.setText("Please wait.");
636: resultArea.setText("The Submitted Results: \n\nName: " + name + "\nNationality: " + nationality + "\nTribal Affiliation: " +
tribe + "\nAge: " + age +
637: "\nGender: " + gender + "\nMarital Status: " + marital + "\nPlace of Birth: " + birthplace + "\nEquipment: " + equipment +
"\nVehicle: " +
638: vehicle + "\nCriminal Record: " + criminalrecord + "\nEducation: " + education + "\nEmployment: " + employment + "\nReligion: " +
religion +
639: "\nSkill: " + skill + "\nAddress: " + address + "\nDOB: " + date);
640:
641: setContentView(resultArea);
642:
643: /* System.out.println("The Submitted Results: \n\nName: " + name + "\nNationality: " + nationality + "\nTribal Affiliation: " +
tribe + "\nAge: " + age +
644: "\nGender: " + gender + "\nMarital Status: " + marital + "\nPlace of Birth: " + birthplace + "\nEquipment: " + equipment +
"\nVehicle: " + vehicle +
645: "\nCriminal Record: " + criminalrecord + "\nEducation: " + education + "\nEmployment: " + employment + "\nReligion: " + religion +
646: "\nSkill: " + skill + "\nAddress: " + address + "\nDOB: " + date);
647: */
648:
649: try {
650: testPersonCreate_json(name, nationality, tribe, gender, age, marital, birthplace, equipment, vehicle, criminalrecord,
education, employment, religion, skill,
651: address, lat, lon, date);
652: } catch (Exception e) {
653: e.printStackTrace();
654: }
655:
656: /*
657: try {
658: Thread.sleep(5000);
659: } catch (InterruptedException e) {
660: e.printStackTrace();
661: }
662: */
663:
664: finish();
665: }//end else
666:
667: }//End sendFeedback
668:
669:
670: public static double convert(double feet) {
671: return feet * feet2meters;
672: }
673:
674:

675: public void testPersonCreate_json(final String name, final String nationality, final String tribe, final String gender,
final int age, final String marital, final String birthplace,
main2.java
676: final String equipment, final String vehicle, final String criminalrecord, final String education, final String employment,
final String religion, final String skill,
677: final String address, final double lat, final double lon, final String date) throws Exception {
678:
679: Thread t = new Thread(){
680:
681: public void run() {
682: //Looper.prepare();
683:
684: SchemeRegistry schemeRegistry = new SchemeRegistry();

```

```

685: schemeRegistry.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));
686: schemeRegistry.register(new Scheme("https", new EasySSLSSocketFactory(), 443));
687:
688: HttpParams params = new BasicHttpParams();
689: params.setParameter(ConnManagerPNames.MAX_TOTAL_CONNECTIONS, 30);
690: params.setParameter(ConnManagerPNames.MAX_CONNECTIONS_PER_ROUTE, new ConnPerRouteBean(30));
691: params.setParameter(HttpProtocolParams.USE_EXPECT_CONTINUE, false);
692: HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
693:
694: ClientConnectionManager cm = new SingleClientConnManager(params, schemeRegistry);
695: HttpClient httpClient = new DefaultHttpClient(cm, params);
696:
697: final ObjectMapper objectMapper = new ObjectMapper();
698:
699: // The top-level JSON object
700: final ObjectNode jsonObj = objectMapper.createObjectNode();
701:
702: // The entities array
703: final ObjectNode entitiesObject = objectMapper.createObjectNode();
704:
705: // The person array
706: ArrayNode personArray = objectMapper.createArrayNode();
707:
708: // The first element in the person array
709: ObjectNode personObject = objectMapper.createObjectNode();
710: ArrayNode namesArray = objectMapper.createArrayNode();
711:
712: //name
713: ObjectNode nameObject = objectMapper.createObjectNode();
714: nameObject.put("fullName", name);
715: namesArray.add(nameObject);
716: personObject.put("names", namesArray);
717:
718: //nationality
719: if (nationality.equals("Choose Suspect Nationality")){
720: //do nothing
721: }
722: else
723: {
724: ObjectNode nationalityObject = objectMapper.createObjectNode();
725: nationalityObject.put("physicalValue", nationality.toUpperCase());
726: personObject.put("nationality", nationalityObject);
727: }
728:
729: //tribe
730: if (tribe.equals("Choose Suspect Tribal Affiliation")){
731: //do nothing
732: }
733: else
734: {
735: personObject.put("ethnicity", tribe.toUpperCase());
736: }
737:
738: //gender
739: if (gender.equals("Choose Suspect Gender")){
740: //do nothing
741: }
742: else
743: {
744: ObjectNode genderObject = objectMapper.createObjectNode();
745: genderObject.put("physicalValue", gender.toUpperCase());
746: personObject.put("gender", genderObject);
747: }
748:
749: //age
750: //String ageStr = Integer.toString(age);


```

main2.java

```

751: personObject.put("age", age);
752:
753: //marital
754: if (marital.equals("Choose Suspect Marital Status")){
755: //do nothing
756: }
757: else
758: {
759: personObject.put("maritalStatus", marital);
760: }
761:
762: //birthplace
763: if (birthplace.equals("Choose Suspect Place of Birth")){
764: //do nothing
765: }
766: else
767: {
768: personObject.put("placeOfBirth", birthplace);
769: }
770:
```

```

771: /*
772: //height
773: ObjectNode heightObject = objectMapper.createObjectNode();
774: heightObject.put("amount", height);
775: personObject.put("height", heightObject);
776:
777: //weight
778: ObjectNode weightObject = objectMapper.createObjectNode();
779: weightObject.put("amount", weight);
780: personObject.put("weight", weightObject);
781: */
782:
783: //DOB
784: personObject.put("dateOfBirth", epoch);
785:
786: //equipment
787: //equipment
788: if (equipment.equals("Choose Suspect Equipment")){
789: //do nothing, but still generate C4ISR OTM Tag
790: ArrayNode remarksArray = objectMapper.createArrayNode();
791: ObjectNode remarksObject = objectMapper.createObjectNode();
792:
793: remarksObject.put("subject","C4ISR OTM");
794: //remarksObject.put("details", equipment);
795:
796: remarksArray.add(remarksObject);
797: personObject.put("remarks", remarksArray);
798: }
799: else
800: {
801: ArrayNode remarksArray = objectMapper.createArrayNode();
802: ObjectNode remarksObject = objectMapper.createObjectNode();
803:
804: remarksObject.put("subject","C4ISR OTM");
805: remarksObject.put("details", equipment);
806:
807: remarksArray.add(remarksObject);
808: personObject.put("remarks", remarksArray);
809: }
810:
811: //vehicle
812: if (vehicle.equals("Choose Suspect Vehicle")){
813: //do nothing
814: }
815: else
816: {
817: personObject.put("description", vehicle);
818: }
819:
820: //criminal record
821: if (criminalrecord.equals("Choose Suspect Criminal Record")){
822: //do nothing
823: }
824: else
825:

```

main2.java

```

826: ArrayNode criminalrecordsArray = objectMapper.createArrayNode();
827: ObjectNode criminalrecordsObject = objectMapper.createObjectNode();
828:
829: criminalrecordsObject.put("verdict", criminalrecord);
830:
831: criminalrecordsArray.add(criminalrecordsObject);
832: personObject.put("criminalRecords", criminalrecordsArray);
833: }
834:
835: //education
836: if (education.equals("Choose Suspect Education")){
837: //do nothing
838: }
839: else
840: {
841: ArrayNode educationArray = objectMapper.createArrayNode();
842: ObjectNode educationObject = objectMapper.createObjectNode();
843:
844: educationObject.put("educationalLevel", education);
845:
846: educationArray.add(educationObject);
847: personObject.put("education", educationArray);
848: }
849:
850: //employment
851: if (employment.equals("Choose Suspect Employment")){
852: //do nothing
853: }
854: else
855: {
856: ArrayNode employmentArray = objectMapper.createArrayNode();

```

```

857: ObjectNode employmentObject = objectMapper.createObjectNode();
858:
859: employmentObject.put("employerType", employment);
860:
861: employmentArray.add(employmentObject);
862: personObject.put("employment", employmentArray);
863: }
864:
865: //religion
866: if (religion.equals("Choose Suspect Religion")){
867: //do nothing
868: }
869: else
870: {
871: ArrayNode religionArray = objectMapper.createArrayNode();
872: ObjectNode religionObject = objectMapper.createObjectNode();
873:
874: religionObject.put("religionName", religion);
875:
876: religionArray.add(religionObject);
877: personObject.put("religions", religionArray);
878: }
879:
880: //skill
881: if (skill.equals("Choose Suspect Skill")){
882: //do nothing
883: }
884: else
885: {
886: ArrayNode skillArray = objectMapper.createArrayNode();
887: ObjectNode skillObject = objectMapper.createObjectNode();
888:
889: skillObject.put("skill", skill);
890:
891: skillArray.add(skillObject);
892: personObject.put("skills", skillArray);
893: }
894:
895: //address
896: if (address.equals("Choose Suspect Address")){
897: //do nothing
898: }
899: else
900: {

main2.java
901: ArrayNode addressArray = objectMapper.createArrayNode();
902: ObjectNode addressObject = objectMapper.createObjectNode();
903:
904: addressObject.put("city", address);
905:
906: addressArray.add(addressObject);
907: personObject.put("addresses", addressArray);
908:
909: //location
910: ArrayNode locationArray = objectMapper.createArrayNode();
911: ArrayNode coordinatesArray = objectMapper.createArrayNode();
912: ObjectNode coordinatesObject = objectMapper.createObjectNode();
913: ObjectNode latlongObject = objectMapper.createObjectNode();
914:
915: coordinatesObject.put("latitude", lat);
916: coordinatesObject.put("longitude", lon);
917: coordinatesObject.put("altitude", 0.0);
918:
919: coordinatesArray.add(coordinatesObject);
920: latlongObject.put("coordinates", coordinatesArray);
921: locationArray.add(latlongObject);
922: latlongObject.put("azimuth", 0.0);
923: latlongObject.put("geometryType", "POINT");
924: personObject.put("locations", locationArray);
925: }
926:
927: // Add the person object to the array.
928: personArray.add(personObject);
929:
930: // Add the person array to the entity object.
931: entitiesObject.put("Person", personArray);
932:
933: // Add the entities object to the top-level JSON object
934: jsonObj.put("entities", entitiesObject);
935:
936: String jsonString = jsonObj.toString();
937:
938: //System.out.println(jsonString);
939:
940: // Set up the client and send the request
941: //String url = "https://ulu.idvrn.arl.army.mil:8443/gg/entity/Person";
942: //String url = "http://pegasus.idvrn.arl.army.mil:8080/gg/entity/Person";

```

```

943:
944: String url=settings.url_all;
945: url=url.concat("entity/Person");
946: //System.out.println("URL=" +url);
947:
948: HttpPost post = new HttpPost(url);
949:
950: post.setHeader("Accept", "application/json");
951: post.setHeader("Content-Type", "application/json");
952: post.setHeader("User-Agent", "Jakarta Commons-HttpClient/3.1");
953:
954: StringEntity se;
955: try {
956: se = new StringEntity(jsonString);
957:
958: se.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
959: post.setEntity(se);
960: } //try
961: catch (UnsupportedEncodingException e1) {
962: e1.printStackTrace();
963: }
964:
965: try{
966: HttpResponse response = httpClient.execute(post);
967:
968: //System.out.println("Status: " + response.getStatusLine().getStatusCode());
969:
970: if (response.getStatusLine().getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
971: if (response.containsHeader("WWW-Authenticate")) {
972: final Header challengeHeader = response.getHeaders("WWW-Authenticate")[0];
973: DigestScheme ds = new DigestScheme();
974: ds.processChallenge(challengeHeader);
975: final Header authHeader = ds.authenticate(

```

main2.java

```

976: new UsernamePasswordCredentials("user2", "ggsecret"),
977: new BasicHttpRequest(HttpPost.METHOD_NAME, url));
978:
979: post.addHeader(authHeader);
980:
981: HttpClient httpClient2 = new DefaultHttpClient(cm, params);
982:
983: response = httpClient2.execute(post);
984:
985: /*
986: System.out.println("Status: " + response.getStatusLine().getStatusCode());
987:
988: HttpResponse end = null;
989: String endResult = null;
990: end = response;
991:
992: BasicResponseHandler myHandler = new BasicResponseHandler();
993:
994: try {
995: endResult = myHandler.handleResponse(end);
996: System.out.println("Result: " + endResult);
997: } catch (HttpResponseException e) {
998: e.printStackTrace();
999: } catch (IOException e) {
1000: e.printStackTrace();
1001: }
1002: */
1003:
1004: httpClient2.getConnectionManager().shutdown();
1005: httpClient.getConnectionManager().shutdown();
1006:
1007: if ( response.getStatusLine().getStatusCode() == HttpStatus.SC_UNSUPPORTED_MEDIA_TYPE) {
1008: System.out.println("Error: " + response.getStatusLine().getStatusCode());
1009: }
1010: }
1011: }
1012: } //try
1013:
1014: catch(Exception e) {
1015: System.out.println("Error " + e);
1016:
1017: } //catch
1018: //Looper.loop();
1019: } //run
1020: }; //thread
1021: t.start();
1022: } //testPerson
1023: } //class

```

```

main3.java
1: package mil.army.arl.hdpt;
2:
3: import java.io.UnsupportedEncodingException;
4: import java.util.Iterator;
5: import org.apache.http.Header;
6: import org.apache.http.HttpResponse;
7: import org.apache.http.HttpStatus;
8: import org.apache.http.HttpVersion;
9: import org.apache.http.auth.UsernamePasswordCredentials;
10: import org.apache.http.client.HttpClient;
11: import org.apache.http.client.methods.HttpPost;
12: import org.apache.http.conn.ClientConnectionManager;
13: import org.apache.http.conn.params.ConnManagerPNames;
14: import org.apache.http.conn.params.ConnPerRouteBean;
15: import org.apache.http.conn.scheme.PlainSocketFactory;
16: import org.apache.http.conn.scheme.Scheme;
17: import org.apache.http.conn.scheme.SchemeRegistry;
18: import org.apache.http.entity.StringEntity;
19: import org.apache.http.impl.auth.DigestScheme;
20: import org.apache.http.impl.client.DefaultHttpClient;
21: import org.apache.http.impl.conn.SingleClientConnManager;
22: import org.apache.http.message.BasicHeader;
23: import org.apache.http.message.BasicHttpRequest;
24: import org.apache.http.params.BasicHttpParams;
25: import org.apache.http.params.HttpParams;
26: import org.apache.http.params.HttpProtocolParams;
27: import org.apache.http.protocol.HTTP;
28: import org.codehaus.jackson.JsonNode;
29: import org.codehaus.jackson.map.ObjectMapper;
30: import org.codehaus.jackson.node.ArrayNode;
31: import org.codehaus.jackson.node.ObjectNode;
32: import android.app.Activity;
33: import android.os.Bundle;
34: import android.view.Gravity;
35: import android.view.View;
36: import android.widget.ArrayAdapter;
37: import android.widget.Spinner;
38: import android.widget.TextView;
39: import android.widget.Toast;
40: import java.text.ParseException;
41: import java.util.Date;
42:
43: import android.app.DatePickerDialog;
44: import android.app.Dialog;
45: import android.widget.Button;
46: import android.widget.DatePicker;
47: import android.widget.EditText;
48:
49:
50: public class main3 extends Activity {
51:
52: private TextView mDateDisplay;
53: private Button mPickDate, mSubmit;
54: private int mYear;
55: private int mMonth;
56: private int mDay;
57: int age;
58: int gender_pos;
59: //int height_pos;
60: //int weight_pos;
61: long epoch;
62: double lat;
63: double lon;
64: long dob;
65:
66: public static final double feet2meters = 0.305;
67: static final int DATE_DIALOG_ID = 0;
68:
69: String uuid=null;
70: String gender=null;
71: String nationality=null;
72: String maritalstatus=null;
73: String placeofbirth=null;
74: String vehicle=null;
75: String equipment=null;

```

```

main3.java
76: String criminalrecord=null;
77: String education=null;
78: String employment=null;
79: String religion=null;
80: String skill=null;
81: String address=null;
82: String tribe=null;
83:
84: JsonNode personNode;

```

```

85:
86: HttpResponse response;
87:
88:
89: public static boolean reset = false;
90:
91: // Called when the activity is first created
92: @Override
93: public void onCreate(Bundle savedInstanceState) {
94:     super.onCreate(savedInstanceState);
95:     setContentView(R.layout.modify);
96:
97:     ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
98:         this, R.array.modifyperson_array, android.R.layout.simple_spinner_item );
99:     adapter1.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
100:
101:
102:     Spinner s1 = (Spinner) findViewById( R.id.modifyperson_spinner );
103:     s1.setAdapter( adapter1 );
104: }
105:
106: // Called when the activity is resumed
107: @Override
108: public void onResume() {
109:     super.onResume();
110:
111:     uuid=null;
112:     gender=null;
113:     nationality=null;
114:     maritalstatus=null;
115:     placeofbirth=null;
116:     vehicle=null;
117:     equipment=null;
118:     criminalrecord=null;
119:     education=null;
120:     employment=null;
121:     religion=null;
122:     skill=null;
123:     address=null;
124:     tribe=null;
125:
126:     personNode = null;
127:     response = null;
128:     //response.getEntity().consumeContent();
129:
130:     setContentView(R.layout.modify);
131:
132:     ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
133:         this, R.array.modifyperson_array, android.R.layout.simple_spinner_item );
134:     adapter1.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
135:
136:
137:     Spinner s1 = (Spinner) findViewById( R.id.modifyperson_spinner );
138:     s1.setAdapter( adapter1 );
139:
140: }
141:
142: public void modifyperson(View button) {
143:     Spinner s1 = (Spinner) findViewById( R.id.modifyperson_spinner );
144:     String person = s1.getSelectedItem().toString();
145:
146:     if(person.equals("Choose Suspect Name"))
147:     {
148:         Toast msg = Toast.makeText(this, "Please Select A Person From The List!", Toast.LENGTH_LONG);
149:         msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
150:
151:         msg.show();

```

main3.java

```

151: }
152:
153: else
154: {
155:     try {
156:         modifyPersonCreate_json(person);
157:     } catch (Exception e) {
158:         e.printStackTrace();
159:     }
160: //else
161: //modifyperson
162:
163:
164: public void modifyPersonCreate_json(final String person){
165:
166:     Thread t = new Thread(){
167:
168:         public void run() {
169:             // Looper.prepare();
170:

```

```

171: SchemeRegistry schemeRegistry = new SchemeRegistry();
172: schemeRegistry.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));
173: schemeRegistry.register(new Scheme("https", new EasySSLocketFactory(), 443));
174:
175: HttpParams params = new BasicHttpParams();
176: params.setParameter(ConnManagerPNames.MAX_TOTAL_CONNECTIONS, 30);
177: params.setParameter(ConnManagerPNames.MAX_CONNECTIONS_PER_ROUTE, new ConnPerRouteBean(30));
178: params.setParameter(HttpProtocolParams.USE_EXPECT_CONTINUE, false);
179: HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
180:
181: ClientConnectionManager cm = new SingleClientConnManager(params, schemeRegistry);
182: HttpClient httpClient2 = new DefaultHttpClient(cm, params);
183:
184: final ObjectMapper objectMapper = new ObjectMapper();
185:
186: ObjectNode jsonObj = objectMapper.createObjectNode();
187: ObjectNode rootObject = objectMapper.createObjectNode();
188: ArrayNode conditionsArray = objectMapper.createArrayNode();
189: ObjectNode conditionsObject = objectMapper.createObjectNode();
190: conditionsObject.put("property", "Person.names.fullName");
191: conditionsObject.put("operator", "LIKE");
192: conditionsObject.put("value", person);
193: conditionsArray.add(conditionsObject);
194: rootObject.put("conditions", conditionsArray);
195: jsonObj.put("maxresults", 200);
196: jsonObj.put("root", rootObject);
197: String jsonString = jsonObj.toString();
198:
199: //System.out.println(jsonString);
200:
201: // Set up the client and send the request
202: //String url = "https://zulu.idvrvn.arl.army.mil:8443/gg/entity/Person";
203: //String url = "http://pegasus.idvrvn.arl.army.mil:8080/gg/entity/Person";
204:
205: String url=settings.url_all;
206: url=url.concat("search");
207: //System.out.println("URL=" +url);
208:
209: HttpPost post = new HttpPost(url);
210:
211: post.setHeader("Accept", "application/json");
212: post.setHeader("Content-Type", "application/json");
213: post.setHeader("User-Agent", "Jakarta Commons-HttpClient/3.1");
214:
215: StringEntity se;
216: try {
217: se = new StringEntity(jsonString);
218:
219: se.setContentEncoding(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
220: post.setEntity(se);
221: } //try
222: catch (UnsupportedEncodingException e1) {
223: e1.printStackTrace();
224: }
225:

```

main3.java

```

226: try{
227:
228: response = httpClient2.execute(post);
229:
230: //System.out.println("Status: " + response.getStatusLine().getStatusCode());
231:
232: if (response.getStatusLine().getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
233: if (response.containsHeader("WWW-Authenticate")) {
234: final Header challengeHeader = response.getHeaders("WWW-Authenticate")[0];
235: DigestScheme ds = new DigestScheme();
236: ds.processChallenge(challengeHeader);
237: final Header authHeader = ds.authenticate(
238: new UsernamePasswordCredentials("user2", "ggsecret"),
239: new BasicHttpRequest(HttpPost.METHOD_NAME, url));
240:
241: post.addHeader(authHeader);
242: //response.getEntity().consumeContent();
243:
244: HttpClient httpClient3 = new DefaultHttpClient(cm, params);
245:
246: response = httpClient3.execute(post);
247:
248: //System.out.println("Status: " + response.getStatusLine().getStatusCode());
249:
250: if ( response.getStatusLine().getStatusCode() == HttpStatus.SC_UNSUPPORTED_MEDIA_TYPE){
251: System.out.println("Error: " + response.getStatusLine().getStatusCode());
252: }
253:
254: //Once you get your response
255:
256: JsonNode rootNode = null;

```

```

257:
258: try{
259: rootNode = objectMapper.readValue(response.getEntity().getContent(), JsonNode.class);
260:
261: }
262: catch(Exception jsonEx){
263: System.out.println("There is a problem with the JSON search response:\n\n" + jsonEx.getStackTrace());
264: }
265:
266: if(rootNode != null)
267: {
268: Iterator<JsonNode> nodes = rootNode.getElements();
269: //System.out.println(rootNode); //this will print out entire json response
270: nodes = (nodes.next()).getElements();
271: JsonNode personArray = nodes.next(); //this hold all the people who match your search, in your case it should just be one
272:
273: personNode = personArray.get(0); //you will want to save this node because you will use it when you send update
274: System.out.println(personNode + "\n\n"); //this will printout the persons entire json structure
275:
276: //ID
277: uuid = personNode.get("id").getTextValue();
278: System.out.println("UUID: " + uuid);
279:
280: //age
281: age = personNode.get("age").asInt();
282: System.out.println("Age: " + age);
283:
284: //gender
285: try {
286: JsonNode genderObject = personNode.get("gender");
287: gender = genderObject.get("physicalValue").getTextValue();
288:
289: if (gender.equals("MALE")){
290: gender = "Male";
291: }
292:
293: if (gender.equals("FEMALE")){
294: gender = "Female";
295: }
296:
297: System.out.println("Gender: " + gender);
298: }
299:
300: catch (Exception e){

main3.java
301: gender = "Choose Suspect Gender";
302: System.out.println("Gender: " + gender);
303: }
304:
305: //marital status
306: try {
307: maritalstatus = personNode.get("maritalStatus").getTextValue();
308:
309: if (maritalstatus.equals("MARRIED")){
310: maritalstatus = "Married";
311: }
312:
313: if (maritalstatus.equals("SINGLE")){
314: maritalstatus = "Single";
315: }
316:
317: System.out.println("Marital Status: " + maritalstatus);
318: }
319:
320: catch (Exception e){
321: maritalstatus = "Choose Suspect Marital Status";
322: System.out.println("Marital Status: " + maritalstatus);
323: }
324:
325: //date of birth
326: try{
327: dob = personNode.get("dateOfBirth").asLong();
328: System.out.println("Date of Birth: " + dob);
329: }
330:
331: catch (Exception e){
332: dob = epoch;
333: System.out.println("Date of Birth: " + dob);
334: }
335:
336: //nationality
337: try{
338: JsonNode nationalityObject = personNode.get("nationality");
339: nationality = nationalityObject.get("physicalValue").getTextValue();
340:
341: if (nationality.equals("MUSLIMA")){
342: nationality = "Arab";

```

```

343: }
344:
345: if (nationality.equals("AFGHAN")){
346: nationality = "Afghan";
347: }
348:
349: System.out.println("Nationality: " + nationality);
350: }
351:
352: catch (Exception e){
353: nationality = "Choose Suspect Nationality";
354: System.out.println("Nationality: " + nationality);
355: }
356:
357: //place of birth
358: try{
359: placeofbirth = personNode.get("placeOfBirth").getTextView();
360:
361: if (placeofbirth.equals("BIA")){
362: placeofbirth = "Born in Area";
363: }
364:
365: if (placeofbirth.equals("BOA")){
366: placeofbirth = "Born outside Area";
367: }
368:
369: System.out.println("Place of Birth: " + placeofbirth);
370: }
371:
372: catch (Exception e){
373: placeofbirth = "Choose Suspect Place of Birth";
374: System.out.println("Place of Birth: " + placeofbirth);
375: }

```

main3.java

```

376:
377: //equipment
378: try{
379: Iterator<JsonNode> ite3 = personNode.get("remarks").getElements();
380: JsonNode temp3 = ite3.next();
381: equipment = temp3.get("details").getTextView();
382: System.out.println("Equipment: " + equipment);
383: }
384:
385: catch (Exception e){
386: equipment = "Choose Suspect Equipment";
387: System.out.println("Equipment: " + equipment);
388: }
389:
390: //vehicle
391: try{
392: vehicle = personNode.get("description").getTextView();
393: System.out.println("Vehicle: " + vehicle);
394: }
395:
396: catch (Exception e){
397: vehicle = "Choose Suspect Vehicle";
398: System.out.println("Vehicle: " + vehicle);
399: }
400:
401: //criminal record
402: try{
403: Iterator<JsonNode> ite4 = personNode.get("criminalRecords").getElements();
404: JsonNode temp4 = ite4.next();
405: criminalrecord = temp4.get("verdict").getTextView();
406:
407: if (criminalrecord.equals("Guilty")){
408: criminalrecord = "Has One";
409: }
410:
411: if (criminalrecord.equals("None")){
412: criminalrecord = "Does Not Have One";
413: }
414:
415: System.out.println("Criminal Records: " + criminalrecord);
416: }
417:
418: catch (Exception e){
419: criminalrecord = "Choose Suspect Criminal Record";
420: System.out.println("Criminal Records: " + criminalrecord);
421: }
422:
423: //education
424: try{
425: Iterator<JsonNode> ite5 = personNode.get("education").getElements();
426: JsonNode temp5 = ite5.next();
427: education = temp5.get("educationalLevel").getTextView();
428: System.out.println("Education: " + education);

```

```

429: }
430:
431: catch (Exception e){
432:     education = "Choose Suspect Education";
433:     System.out.println("Education: " + education);
434: }
435:
436: //employment
437: try{
438:     Iterator<JsonNode> ite6 = personNode.get("employment").getElements();
439:     JsonNode temp6 = ite6.next();
440:     employment = temp6.get("employerType").getTextValue();
441:
442:     if (employment.equals("NE")){
443:         employment = "Not Employed";
444:     }
445:
446:     if (employment.equals("WC")){
447:         employment = "White Collar";
448:     }
449:

450:     if (employment.equals("BC")) {

main3.java
451:     employment = "Blue Collar";
452: }
453:
454: System.out.println("Employment: " + employment);
455: }
456:
457: catch (Exception e){
458:     employment = "Choose Suspect Employment";
459:     System.out.println("Employment: " + employment);
460: }
461:
462: //religion
463: try{
464:     Iterator<JsonNode> ite7 = personNode.get("religions").getElements();
465:     JsonNode temp7 = ite7.next();
466:     religion = temp7.get("religionName").getTextValue();
467:
468:     if (religion.equals("Mld")){
469:         religion = "Mild Theology";
470:     }
471:
472:     if (religion.equals("Rad")){
473:         religion = "Radical Theology";
474:     }
475:
476:     System.out.println("Religion: " + religion);
477: }
478:
479: catch (Exception e){
480:     religion = "Choose Suspect Religion";
481:     System.out.println("Religion: " + religion);
482: }
483:
484: //skill
485: try{
486:     Iterator<JsonNode> ite8 = personNode.get("skills").getElements();
487:     JsonNode temp8 = ite8.next();
488:     skill = temp8.get("skill").getTextValue();
489:
490:     if (skill.equals("PH")){
491:         skill = "Photography";
492:     }
493:
494:     if (skill.equals("WR")){
495:         skill = "Writing";
496:     }
497:
498:     if (skill.equals("EL")){
499:         skill = "Electrical";
500:     }
501:
502:     if (skill.equals("ME")){
503:         skill = "Mechanical";
504:     }
505:
506:     if (skill.equals("CO")){
507:         skill = "Computer";
508:     }
509:
510:     if (skill.equals("DR")){
511:         skill = "Driving";
512:     }
513:
514:     if (skill.equals("FI")){

```

```

515: skill = "Financial";
516: }
517:
518: System.out.println("Skill: " + skill);
519: }
520:
521: catch (Exception e){
522: skill = "Choose Suspect Skill";
523: System.out.println("Skill: " + skill);
524: }

525:

main3.java
526: //address
527: try{
528: Iterator<JsonNode> ite9 = personNode.get("addresses").getElements();
529: JsonNode temp9 = ite9.next();
530: address = temp9.get("city").getTextValue();
531:
532: if (address.equals("TSV")){
533: address = "Times Square Village";
534: lat = 39.983611;
535: lon = -74.43;
536: }
537:
538: if (address.equals("VNV")){
539: address = "Viet Nam Village";
540: lat = 39.97883;
541: lon = -74.43;
542: }
543:
544: if (address.equals("VV")){
545: address = "Vertol Village";
546: lat = 39.97278;
547: lon = -74.4275;
548: }
549:
550: if (address.equals("HAV")){
551: address = "Hanover Village";
552: lat = 40.00944;
553: lon = -74.5575;
554: }
555:
556: if (address.equals("CCV")){
557: address = "Cook Corner Village";
558: lat = 40.01556;
559: lon = -74.5525;
560: }
561:
562: if (address.equals("GT")){
563: address = "Gredge Town";
564: lat = 40.03083;
565: lon = -74.52444;
566: }
567:
568: if (address.equals("UV")){
569: address = "Utes Village";
570: lat = 40.04583;
571: lon = -74.46056;
572: }
573:
574: if (address.equals("HOV")){
575: address = "Horizon Village";
576: lat = 40.00667;
577: lon = -74.45111;
578: }
579:
580: System.out.println("Address: " + address);
581: }
582:
583: catch (Exception e){
584: address = "Choose Suspect Address";
585: System.out.println("Address: " + address);
586: }
587:
588: //tribe
589: try{
590: tribe = personNode.get("ethnicity").getTextValue();
591:
592: if (tribe.equals("PASHTU")){
593: tribe = "Pashtu";
594: }
595:
596: if (tribe.equals("BALOCH")){
597: tribe = "Baloch";

```

```

598: }
599:

600: if (tribe.equals("HAZARA")) {

main3.java
601: tribe = "Hazara";
602: }
603:
604: if (tribe.equals("TAJIK")){
605: tribe = "Tajik";
606: }
607:
608: System.out.println("Tribal Affiliation: " + tribe);
609: }
610:
611: catch (Exception e){
612: tribe = "Choose Suspect Tribal Affiliation";
613: System.out.println("Tribal Affiliation: " + tribe);
614: }
615:
616: //After getting uuid clear out the response
617: //EntityUtils.consume(response.getEntity());
618: //response.getEntity().consumeContent();
619:
620: }
621:
622: httpClient3.getConnectionManager().shutdown();
623: httpClient2.getConnectionManager().shutdown();
624: }
625: }
626:
627:
628: }//try
629:
630: catch(Exception e) {
631: //System.out.println("Error1 " + e.getCause());
632: //e.printStackTrace();
633: httpClient2.getConnectionManager().shutdown();
634:
635: }//catch
636:
637: // Looper.loop();
638: }//run
639: }////thread
640: t.start();
641:
642: // Looper.myLooper().quit();
643:
644: while (t.isAlive()){
645: try {
646: //System.out.println("Thread Alive");
647: t.join(2000);
648: } catch (InterruptedException e) {
649: e.printStackTrace();
650: }
651: }
652:
653: setContentView(R.layout.form);
654:
655: if(nationality == (null) && gender == (null) && maritalstatus == (null)
656: && placeofbirth == (null) && equipment == (null) && vehicle == (null)
657: && criminalrecord == (null) && education == (null) && employment == (null)
658: && religion == (null) && skill == (null) && address == (null) && tribe == (null))
659: {
660: Toast msg = Toast.makeText(this, "Person not found in the Global Graph. Creating a new person: " + person + ".",
Toast.LENGTH_LONG);
661: msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
662: msg.show();
663: }
664:
665: // capture our View elements
666: mDateDisplay = (TextView) findViewById(R.id.dateDisplay);
667: mPickDate = (Button) findViewById(R.id.pickDate);
668: mSubmit = (Button) findViewById(R.id.ButtonSendFeedback);
669:
670: // add a click listener to the button
671: mSubmit.setOnClickListener(new View.OnClickListener() {
672: public void onClick(View x) {
673: sendFeedback(x);
674: }
675: });

```

```

main3.java
676:
677: // add a click listener to the button
678: mPickDate.setOnClickListener(new View.OnClickListener() {
679:     public void onClick(View v) {
680:         showDialog(DATE_DIALOG_ID);
681:     }
682: });
683:
684: // get the current date
685: final Calendar c = Calendar.getInstance();
686: mYear = c.get(Calendar.YEAR);
687: mMonth = c.get(Calendar.MONTH);
688: mDay = c.get(Calendar.DAY_OF_MONTH);
689:
690: // display the current date (this method is below)
691: updateDisplay();
692:
693: Date birthday = convertDOB(dob);
694:
695: mYear = birthday.getYear()+1900;
696: mMonth = birthday.getMonth();
697: mDay = birthday.getDay();
698:
699: //System.out.println(birthday);
700: updateDisplay();
701:
702: final EditText nameField = (EditText) findViewById(R.id.EditTextName);
703: nameField.setText(person);
704:
705: ArrayAdapter<CharSequence> adapter0 = ArrayAdapter.createFromResource(
706:     this, R.array.nationality_array, android.R.layout.simple_spinner_item );
707: adapter0.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
708:
709: Spinner s0 = (Spinner) findViewById( R.id.nationality_spinner );
710:
711: s0.setAdapter( adapter0 );
712:
713: int spinnerPosition0 = adapter0.getPosition(nationality);
714:
715: //set the default according to value
716: s0.setSelection(spinnerPosition0);
717:
718: ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
719:     this, R.array.gender_array, android.R.layout.simple_spinner_item );
720: adapter1.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
721:
722: Spinner s1 = (Spinner) findViewById( R.id.gender_spinner );
723: s1.setAdapter( adapter1 );
724:
725: s1.setAdapter( adapter1 );
726:
727: int spinnerPosition1 = adapter1.getPosition(gender);
728: //set the default according to value
729: s1.setSelection(spinnerPosition1);
730:
731: /*
732: ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(
733:     this, R.array.height_array, android.R.layout.simple_spinner_item );
734: adapter2.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
735:
736:
737: Spinner s2 = (Spinner) findViewById( R.id.height_spinner );
738: s2.setAdapter( adapter2 );
739: */
740:
741: ArrayAdapter<CharSequence> adapter21 = ArrayAdapter.createFromResource(
742:     this, R.array.marital_array, android.R.layout.simple_spinner_item );
743: adapter21.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
744:
745: Spinner s21 = (Spinner) findViewById( R.id.marital_spinner );
746:
747: s21.setAdapter( adapter21 );
748:
749: int spinnerPosition21 = adapter21.getPosition(maritalstatus);

750: //set the default according to value

```

```

main3.java
751: s21.setSelection(spinnerPosition21);
752:
753: ArrayAdapter<CharSequence> adapter22 = ArrayAdapter.createFromResource(
754:     this, R.array.birthplace_array, android.R.layout.simple_spinner_item );
755: adapter22.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
756:
757: Spinner s22 = (Spinner) findViewById( R.id.birthplace_spinner );
758:
759: s22.setAdapter( adapter22 );

```

```

760:
761: int spinnerPosition22 = adapter22.getPosition(placeofbirth);
762: //set the default according to value
763: s22.setSelection(spinnerPosition22);
764:
765: /*
766: ArrayAdapter<CharSequence> adapter3 = ArrayAdapter.createFromResource(
767: this, R.array.weight_array, android.R.layout.simple_spinner_item );
768: adapter3.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
769:
770: Spinner s3 = (Spinner) findViewById( R.id.weight_spinner );
771: s3.setAdapter( adapter3 );
772: */
773:
774: ArrayAdapter<CharSequence> adapter4 = ArrayAdapter.createFromResource(
775: this, R.array.equipment_array, android.R.layout.simple_spinner_item );
776: adapter4.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
777:
778: Spinner s4 = (Spinner) findViewById( R.id.equipment_spinner );
779:
780: s4.setAdapter( adapter4 );
781:
782: int spinnerPosition4 = adapter4.getPosition(equipment);
783: //set the default according to value
784: s4.setSelection(spinnerPosition4);
785:
786: ArrayAdapter<CharSequence> adapter5 = ArrayAdapter.createFromResource(
787: this, R.array.vehicle_array, android.R.layout.simple_spinner_item );
788: adapter5.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
789:
790: Spinner s5 = (Spinner) findViewById( R.id.vehicle_spinner );
791:
792: s5.setAdapter( adapter5 );
793:
794: int spinnerPosition5 = adapter5.getPosition(vehicle);
795: //set the default according to value
796: s5.setSelection(spinnerPosition5);
797:
798: ArrayAdapter<CharSequence> adapter6 = ArrayAdapter.createFromResource(
799: this, R.array.criminalrecord_array, android.R.layout.simple_spinner_item );
800: adapter6.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
801:
802: Spinner s6 = (Spinner) findViewById( R.id.criminalrecord_spinner );
803:
804: s6.setAdapter( adapter6 );
805:
806: int spinnerPosition6 = adapter6.getPosition(criminalrecord);
807: //set the default according to value
808: s6.setSelection(spinnerPosition6);
809:
810: ArrayAdapter<CharSequence> adapter7 = ArrayAdapter.createFromResource(
811: this, R.array.education_array, android.R.layout.simple_spinner_item );
812: adapter7.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
813:
814: Spinner s7 = (Spinner) findViewById( R.id.education_spinner );
815:
816: s7.setAdapter( adapter7 );
817:
818: int spinnerPosition7 = adapter7.getPosition(education);
819: //set the default according to value
820: s7.setSelection(spinnerPosition7);
821:
822: ArrayAdapter<CharSequence> adapter8 = ArrayAdapter.createFromResource(
823: this, R.array.employment_array, android.R.layout.simple_spinner_item );
824: adapter8.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );

825:

main3.java
826: Spinner s8 = (Spinner) findViewById( R.id.employment_spinner );
827:
828: s8.setAdapter( adapter8 );
829:
830: int spinnerPosition8 = adapter8.getPosition(employment);
831: //set the default according to value
832: s8.setSelection(spinnerPosition8);
833:
834: ArrayAdapter<CharSequence> adapter9 = ArrayAdapter.createFromResource(
835: this, R.array.religion_array, android.R.layout.simple_spinner_item );
836: adapter9.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
837:
838: Spinner s9 = (Spinner) findViewById( R.id.religion_spinner );
839:
840: s9.setAdapter( adapter9 );
841:
842: int spinnerPosition9 = adapter9.getPosition(religion);
843: //set the default according to value
844: s9.setSelection(spinnerPosition9);
845:
```

```

846: ArrayAdapter<CharSequence> adapter10 = ArrayAdapter.createFromResource(
847:     this, R.array.skill_array, android.R.layout.simple_spinner_item );
848: adapter10.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
849:
850: Spinner s10 = (Spinner) findViewById( R.id.skill_spinner );
851:
852: s10.setAdapter( adapter10 );
853:
854: int spinnerPosition10 = adapter10.getPosition(skill);
855: //set the default according to value
856: s10.setSelection(spinnerPosition10);
857:
858: ArrayAdapter<CharSequence> adapter11 = ArrayAdapter.createFromResource(
859:     this, R.array.address_array, android.R.layout.simple_spinner_item );
860: adapter11.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
861:
862: Spinner s11 = (Spinner) findViewById( R.id.address_spinner );
863:
864: s11.setAdapter( adapter11 );
865:
866: int spinnerPosition11 = adapter11.getPosition(address);
867: //set the default according to value
868: s11.setSelection(spinnerPosition11);
869:
870: ArrayAdapter<CharSequence> adapter12 = ArrayAdapter.createFromResource(
871:     this, R.array.tribe_array, android.R.layout.simple_spinner_item );
872: adapter12.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );
873:
874: Spinner s12 = (Spinner) findViewById( R.id.tribe_spinner );
875:
876: s12.setAdapter( adapter12 );
877:
878: int spinnerPosition12 = adapter12.getPosition(tribe);
879: //set the default according to value
880: s12.setSelection(spinnerPosition12);
881:
882: }//modify person
883:
884: public Date convertDOB(long epoch) {
885:
886: //long epoch = Long.parseLong( epoch );
887: Date expiry = new Date( epoch );
888: return expiry;
889: }
890:
891: // updates the date in the TextView
892: private void updateDisplay() {
893: mDateDisplay.setText(
894: new StringBuilder()
895: // Month is 0 based so add 1
896: .append(mMonth + 1).append("-")
897: .append(mDay).append("-")
898: .append(mYear).append(" "));
899: try {
900: String nMonth = String.format("%02d", (mMonth+1));

```

main3.java

```

901: String nDay = String.format("%02d", mDay);
902: String eDate = (nMonth + "/" + nDay + "/" + mYear);
903:
904: epoch = new java.text.SimpleDateFormat("MM/dd/yyyy").parse(eDate).getTime();
905:
906:
907: Date now = new Date();
908: int month = mMonth;
909: int day = mDay;
910: int year = mYear;
911:
912: int nowMonth = now.getMonth()+1;
913: int nowYear = now.getYear()+1900;
914: int result = nowYear - year;
915:
916: if (month > nowMonth) {
917: result--;
918: }
919: else if (month == nowMonth) {
920: int nowDay = now.getDate();
921:
922: if (day > nowDay) {
923: result--;
924: }
925: }
926: age = result;
927:
928: } catch (ParseException e) {
929: e.printStackTrace();
930: }
931: }//updateDisplay

```

```

932:
933:
934: // the callback received when the user "sets" the date in the dialog
935: private DatePickerDialog.OnDateSetListener mDateSetListener =
936: new DatePickerDialog.OnDateSetListener() {
937:
938: public void onDateSet(DatePicker view, int year,
939: int monthOfYear, int dayOfMonth) {
940: mYear = year;
941: mMonth = monthOfYear;
942: mDay = dayOfMonth;
943: updateDisplay();
944: }
945: };
946:
947: @Override
948: protected Dialog onCreateDialog(int id) {
949: switch (id) {
950: case DATE_DIALOG_ID:
951: return new DatePickerDialog(this,
952: mDateSetListener,
953: mYear, mMonth, mDay);
954: }
955: return null;
956: }
957:
958: public void sendFeedback(View button) {
959:
960: final EditText nameField = (EditText) findViewById(R.id.EditTextName);
961: String name = nameField.getText().toString();
962:
963: Spinner s = (Spinner) findViewById( R.id.nationality_spinner );
964: String nationality = s.getSelectedItem().toString();
965:
966: Spinner s2 = (Spinner) findViewById( R.id.gender_spinner );
967: String gender = s2.getSelectedItem().toString();
968:
969: Spinner s21 = (Spinner) findViewById( R.id.marital_spinner );
970: String marital = s21.getSelectedItem().toString();
971:
972: Spinner s22 = (Spinner) findViewById( R.id.birthplace_spinner );
973: String birthplace = s22.getSelectedItem().toString();
974:
975: /

```

main3.java

```

976: Spinner s3 = (Spinner) findViewById( R.id.height_spinner );
977: String height = s3.getSelectedItem().toString();
978:
979: Spinner s4 = (Spinner) findViewById( R.id.weight_spinner );
980: String weight = s4.getSelectedItem().toString();
981: */
982:
983: Spinner s4 = (Spinner) findViewById( R.id.equipment_spinner );
984: String equipment = s4.getSelectedItem().toString();
985:
986: Spinner s5 = (Spinner) findViewById( R.id.vehicle_spinner );
987: String vehicle = s5.getSelectedItem().toString();
988:
989: Spinner s6 = (Spinner) findViewById( R.id.criminalrecord_spinner );
990: String criminalrecord = s6.getSelectedItem().toString();
991:
992: Spinner s7 = (Spinner) findViewById( R.id.education_spinner );
993: String education = s7.getSelectedItem().toString();
994:
995: Spinner s8 = (Spinner) findViewById( R.id.employment_spinner );
996: String employment = s8.getSelectedItem().toString();
997:
998: Spinner s9 = (Spinner) findViewById( R.id.religion_spinner );
999: String religion = s9.getSelectedItem().toString();
1000:
1001: Spinner s10 = (Spinner) findViewById( R.id.skill_spinner );
1002: String skill = s10.getSelectedItem().toString();
1003:
1004: Spinner s11 = (Spinner) findViewById( R.id.address_spinner );
1005: String address = s11.getSelectedItem().toString();
1006:
1007: Spinner s12 = (Spinner) findViewById( R.id.tribe_spinner );
1008: String tribe = s12.getSelectedItem().toString();
1009:
1010: final TextView dateField = (TextView) findViewById(R.id.dateDisplay);
1011: String date = dateField.getText().toString();
1012:
1013: if(false)
1014: {
1015: //do nothing
1016: }
1017:

```

```

1018: /*
1019: if(name.isEmpty() || nationality.equals("Choose Suspect Nationality") || gender.equals("Choose Suspect Gender") ||
marital.equals("Choose Suspect Marital Status")
1020: || birthplace.equals("Choose Suspect Place of Birth") || equipment.equals("Choose Suspect Equipment") ||
vehicle.equals("Choose Suspect Vehicle")
1021: || criminalrecord.equals("Choose Suspect Criminal Record") || education.equals("Choose Suspect Education") ||
employment.equals("Choose Suspect Employment")
1022: || religion.equals("Choose Suspect Religion") || skill.equals("Choose Suspect Skill") || address.equals("Choose Suspect
Address")
1023: || tribe.equals("Choose Suspect Tribal Affiliation") )
1024: {
1025: Toast msg = Toast.makeText(this, "Please enter information for all fields!", Toast.LENGTH_LONG);
1026: msg.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
1027: msg.show();
1028: }
1029: */
1030:
1031: else {
1032: //double oHeight = Double.parseDouble( height );
1033: //double mHeight = convert(oHeight);
1034: //String nHeight = Double.toString(mHeight);
1035: double lat = 0, lon = 0;
1036:
1037: if (nationality.equals("Arab")){
1038: nationality = "Muslma";
1039: }
1040:
1041: if (birthplace.equals("Born in Area")){
1042: birthplace = "BIA";
1043: }
1044:
1045: if (birthplace.equals("Born outside Area")){
1046: birthplace = "BOA";
1047: }
1048:
1049: if (criminalrecord.equals("Has One")){
1050: criminalrecord = "Guilty";

```

main3.java

```

1051: }
1052:
1053: if (criminalrecord.equals("Does Not Have One")){
1054: criminalrecord = "None";
1055: }
1056:
1057: if (employment.equals("Not Employed")){
1058: employment = "NE";
1059: }
1060:
1061: if (employment.equals("White Collar")){
1062: employment = "WC";
1063: }
1064:
1065: if (employment.equals("Blue Collar")){
1066: employment = "BC";
1067: }
1068:
1069: if (religion.equals("Mild Theology")){
1070: religion = "Mld";
1071: }
1072:
1073: if (religion.equals("Radical Theology")){
1074: religion = "Rad";
1075: }
1076:
1077: if (skill.equals("Photography")){
1078: skill = "PH";
1079: }
1080:
1081: if (skill.equals("Writing")){
1082: skill = "WR";
1083: }
1084:
1085: if (skill.equals("Electrical")){
1086: skill = "EL";
1087: }
1088:
1089: if (skill.equals("Mechanical")){
1090: skill = "ME";
1091: }
1092:
1093: if (skill.equals("Computer")){
1094: skill = "CO";
1095: }
1096:
1097: if (skill.equals("Driving")){
1098: skill = "DR";
1099: }

```

```

1100:
1101: if (skill.equals("Financial")){
1102: skill = "FI";
1103: }
1104:
1105: if (address.equals("Times Square Village")){
1106: address = "TSV";
1107: lat = 39.983611;
1108: lon = -74.43;
1109: }
1110:
1111: if (address.equals("Viet Nam Village")){
1112: address = "VNV";
1113: lat = 39.97883;
1114: lon = -74.43;
1115: }
1116:
1117: if (address.equals("Vertol Village")){
1118: address = "VV";
1119: lat = 39.97278;
1120: lon = -74.4275;
1121: }
1122:
1123: if (address.equals("Hanover Village")){
1124: address = "HAV";
1125: lat = 40.00944;

main3.java
1126: lon = -74.5575;
1127: }
1128:
1129: if (address.equals("Cook Corner Village")){
1130: address = "CCV";
1131: lat = 40.01556;
1132: lon = -74.5525;
1133: }
1134:
1135: if (address.equals("Gredge Town")){
1136: address = "GT";
1137: lat = 40.03083;
1138: lon = -74.52444;
1139: }
1140:
1141: if (address.equals("Utes Village")){
1142: address = "UV";
1143: lat = 40.04583;
1144: lon = -74.46056;
1145: }
1146:
1147: if (address.equals("Horizon Village")){
1148: address = "HOV";
1149: lat = 40.00667;
1150: lon = -74.45111;
1151: }
1152:
1153: TextView resultArea;
1154: resultArea = new TextView(this);
1155: resultArea.setText("Please wait.");
1156: resultArea.setText("The Submitted Results: \n\nName: " + name + "\nNationality: " + nationality + "\nTribal Affiliation: " +
tribe + "\nAge: " + age + "\nGender: " + gender +
1157: "\nMarital Status: " + marital + "\nPlace of Birth: " + birthplace + "\nEquipment: " + equipment + "\nVehicle: " + vehicle +
1158: "\nCriminal Record: " + criminalrecord + "\nEducation: " + education + "\nEmployment: " + employment + "\nReligion: " +
religion +
1159: "\nSkill: " + skill + "\nAddress: " + address + "\nDOB: " + date);
1160:
1161: setContentView(resultArea);
1162:
1163: /* System.out.println("The Submitted Results: \n\nName: " + name + "\nNationality: " + nationality + "\nTribal Affiliation: " +
tribe + "\nAge: " + age +
1164: "\nGender: " + gender + "\nMarital Status: " + marital + "\nPlace of Birth: " + birthplace + "\nEquipment: " + equipment +
"\nVehicle: " + vehicle +
1165: "\nCriminal Record: " + criminalrecord + "\nEducation: " + education + "\nEmployment: " + employment + "\nReligion: " +
religion +
1166: "\nSkill: " + skill + "\nAddress: " + address + "\nDOB: " + date);
1167: */
1168:
1169: try {
1170: testPersonCreate_json(name, nationality, tribe, gender, age, marital, birthplace, equipment, vehicle, criminalrecord,
education, employment, religion, skill, address,
1171: lat, lon, date);
1172: } catch (Exception e) {
1173: e.printStackTrace();
1174: }
1175:
1176: /*
1177: try {
1178: Thread.sleep(5000);
1179: } catch (InterruptedException e) {

```

```

1180: e.printStackTrace();
1181: }
1182: */
1183:
1184: finish();
1185: }//end else
1186:
1187: }//End sendFeedback
1188:
1189:
1190: public double convert(double feet) {
1191: return feet * feet2meters;
1192: }
1193:
1194:
1195: public void testPersonCreate_json(final String name, final String nationality, final String tribe, final String gender,
1196: final int age, final String marital, final String birthplace,
1197: final String equipment, final String vehicle, final String criminalrecord, final String education, final String employment,
1198: final String religion, final String skill,
1199: final String address, final double lat, final double lon, final String date) throws Exception {
1200:
1201: Thread t = new Thread(){
1202:
1203:
1204: SchemeRegistry schemeRegistry = new SchemeRegistry();
1205: schemeRegistry.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));
1206: schemeRegistry.register(new Scheme("https", new EasySSLSocketFactory(), 443));
1207:
1208: HttpParams params = new BasicHttpParams();
1209: params.setParameter(ConnManagerPNames.MAX_TOTAL_CONNECTIONS, 30);
1210: params.setParameter(ConnManagerPNames.MAX_CONNECTIONS_PER_ROUTE, new ConnPerRouteBean(30));
1211: params.setParameter(HttpProtocolParams.USE_EXPECT_CONTINUE, false);
1212: HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
1213:
1214: ClientConnectionManager cm = new SingleClientConnManager(params, schemeRegistry);
1215: HttpClient httpClient4 = new DefaultHttpClient(cm, params);
1216:
1217: final ObjectMapper objectMapper = new ObjectMapper();
1218:
1219: // The top-level JSON object
1220: final ObjectNode jsonObj = objectMapper.createObjectNode();
1221:
1222: // The entities array
1223: final ObjectNode entitiesObject = objectMapper.createObjectNode();
1224:
1225: // The person array
1226: ArrayNode personArray = objectMapper.createArrayNode();
1227:
1228: // The first element in the person array
1229: ObjectNode personObject = objectMapper.createObjectNode();
1230: ArrayNode namesArray = objectMapper.createArrayNode();
1231:
1232: //name
1233: ObjectNode nameObject = objectMapper.createObjectNode();
1234: nameObject.put("fullName", name);
1235: namesArray.add(nameObject);
1236: personObject.put("names", namesArray);
1237:
1238: //nationality
1239: if (nationality.equals("Choose Suspect Nationality")){
1240: //do nothing
1241: }
1242: else
1243: {
1244: ObjectNode nationalityObject = objectMapper.createObjectNode();
1245: nationalityObject.put("physicalValue", nationality.toUpperCase());
1246: personObject.put("nationality", nationalityObject);
1247: }
1248:
1249: //tribe
1250: if (tribe.equals("Choose Suspect Tribal Affiliation")){
1251: //do nothing
1252: }
1253: else
1254: {
1255: personObject.put("ethnicity", tribe.toUpperCase());
1256: }
1257:
1258: //gender
1259: if (gender.equals("Choose Suspect Gender")){
1260: //do nothing
1261: }
1262: else
1263: {

```

```

1264: ObjectNode genderObject = objectMapper.createObjectNode();
1265: genderObject.put("physicalValue", gender.toUpperCase());
1266: personObject.put("gender", genderObject);
1267: }
1268:
1269: //age
1270: //String ageStr = Integer.toString(age);
1271: personObject.put("age", age);
1272:
1273: //marital
1274: if (marital.equals("Choose Suspect Marital Status")){
1275: //do nothing

main3.java
1276: }
1277: else
1278: {
1279: personObject.put("maritalStatus", marital);
1280: }
1281:
1282: //birthplace
1283: if (birthplace.equals("Choose Suspect Place of Birth")){
1284: //do nothing
1285: }
1286: else
1287: {
1288: personObject.put("placeOfBirth", birthplace);
1289: }
1290:
1291: /*
1292: //height
1293: ObjectNode heightObject = objectMapper.createObjectNode();
1294: heightObject.put("amount", height);
1295: personObject.put("height", heightObject);
1296:
1297: //weight
1298: ObjectNode weightObject = objectMapper.createObjectNode();
1299: weightObject.put("amount", weight);
1300: personObject.put("weight", weightObject);
1301: */
1302:
1303: //DOB
1304: personObject.put("dateOfBirth", epoch);
1305:
1306: //equipment
1307: if (equipment.equals("Choose Suspect Equipment")){
1308: //do nothing, but still generate C4ISR OTM Tag
1309: ArrayNode remarksArray = objectMapper.createArrayNode();
1310: ObjectNode remarksObject = objectMapper.createObjectNode();
1311:
1312: remarksObject.put("subject", "C4ISR OTM");
1313: //remarksObject.put("details", equipment);
1314:
1315: remarksArray.add(remarksObject);
1316: personObject.put("remarks", remarksArray);
1317: }
1318: else
1319: {
1320: ArrayNode remarksArray = objectMapper.createArrayNode();
1321: ObjectNode remarksObject = objectMapper.createObjectNode();
1322:
1323: remarksObject.put("subject", "C4ISR OTM");
1324: remarksObject.put("details", equipment);
1325:
1326: remarksArray.add(remarksObject);
1327: personObject.put("remarks", remarksArray);
1328: }
1329:
1330: //vehicle
1331: if (vehicle.equals("Choose Suspect Vehicle")){
1332: //do nothing
1333: }
1334: else
1335: {
1336: personObject.put("description", vehicle);
1337: }
1338:
1339: //criminal record
1340: if (criminalrecord.equals("Choose Suspect Criminal Record")){
1341: //do nothing
1342: }
1343: else
1344: {
1345: ArrayNode criminalrecordsArray = objectMapper.createArrayNode();
1346: ObjectNode criminalrecordsObject = objectMapper.createObjectNode();

```

```

1347:
1348: criminalrecordsObject.put("verdict", criminalrecord);
1349:

1350: criminalrecordsArray.add(criminalrecordsObject);

main3.java
1351: personObject.put("criminalRecords", criminalrecordsArray);
1352: }
1353:
1354: //education
1355: if (education.equals("Choose Suspect Education")){
1356: //do nothing
1357: }
1358: else
1359: {
1360: ArrayNode educationArray = objectMapper.createArrayNode();
1361: ObjectNode educationObject = objectMapper.createObjectNode();
1362:
1363: educationObject.put("educationalLevel", education);
1364:
1365: educationArray.add(educationObject);
1366: personObject.put("education", educationArray);
1367: }
1368:
1369: //employment
1370: if (employment.equals("Choose Suspect Employment")){
1371: //do nothing
1372: }
1373: else
1374: {
1375: ArrayNode employmentArray = objectMapper.createArrayNode();
1376: ObjectNode employmentObject = objectMapper.createObjectNode();
1377:
1378: employmentObject.put("employerType", employment);
1379:
1380: employmentArray.add(employmentObject);
1381: personObject.put("employment", employmentArray);
1382: }
1383:
1384: //religion
1385: if (religion.equals("Choose Suspect Religion")){
1386: //do nothing
1387: }
1388: else
1389: {
1390: ArrayNode religionArray = objectMapper.createArrayNode();
1391: ObjectNode religionObject = objectMapper.createObjectNode();
1392:
1393: religionObject.put("religionName", religion);
1394:
1395: religionArray.add(religionObject);
1396: personObject.put("religions", religionArray);
1397: }
1398:
1399: //skill
1400: if (skill.equals("Choose Suspect Skill")){
1401: //do nothing
1402: }
1403: else
1404: {
1405: ArrayNode skillArray = objectMapper.createArrayNode();
1406: ObjectNode skillObject = objectMapper.createObjectNode();
1407:
1408: skillObject.put("skill", skill);
1409:
1410: skillArray.add(skillObject);
1411: personObject.put("skills", skillArray);
1412: }
1413:
1414: //address
1415: if (address.equals("Choose Suspect Address")){
1416: //do nothing
1417: }
1418: else
1419: {
1420: ArrayNode addressArray = objectMapper.createArrayNode();
1421: ObjectNode addressObject = objectMapper.createObjectNode();
1422:
1423: addressObject.put("city", address);
1424:
1425: addressArray.add(addressObject);

```

```

main3.java
1426: personObject.put("addresses", addressArray);
1427:
1428: //location
1429: ArrayNode locationArray = objectMapper.createArrayNode();
1430: ArrayNode coordinatesArray = objectMapper.createArrayNode();
1431: ObjectNode coordinatesObject = objectMapper.createObjectNode();
1432: ObjectNode latlongObject = objectMapper.createObjectNode();
1433:
1434: coordinatesObject.put("latitude", lat);
1435: coordinatesObject.put("longitude", lon);
1436: coordinatesObject.put("altitude", 0.0);
1437:
1438: coordinatesArray.add(coordinatesObject);
1439: latlongObject.put("coordinates", coordinatesArray);
1440: locationArray.add(latlongObject);
1441: latlongObject.put("azimuth", 0.0);
1442: latlongObject.put("geometryType", "POINT");
1443: personObject.put("locations", locationArray);
1444: }
1445:
1446: //uuid
1447: personObject.put("id", uuid);
1448:
1449: // Add the person object to the array.
1450: personArray.add(personObject);
1451:
1452: // Add the person array to the entity object.
1453: entitiesObject.put("Person", personArray);
1454:
1455: // Add the entities object to the top-level JSON object
1456: jsonObj.put("entities", entitiesObject);
1457:
1458: String jsonString = jsonObj.toString();
1459:
1460: //System.out.println(personNode);
1461:
1462: System.out.println(jsonString);
1463:
1464: //String jsonString2 = personNode.toString();
1465: //jsonString = jsonString2;
1466:
1467: // Set up the client and send the request
1468: //String url = "https://zulu.idvrn.arl.army.mil:8443/gg/entity/Person";
1469: //String url = "http://pegasus.idvrn.arl.army.mil:8080/gg/entity/Person";
1470:
1471: String url=settings.url_all;
1472: url=url.concat("entity/Person/" + uuid);
1473: //System.out.println("URL=" +url);
1474:
1475: HttpPost post = new HttpPost(url);
1476:
1477: post.setHeader("Accept", "application/json");
1478: post.setHeader("Content-Type", "application/json");
1479: post.setHeader("User-Agent", "Jakarta Commons-HttpClient/3.1");
1480:
1481: StringEntity se;
1482: try {
1483: se = new StringEntity(jsonString);
1484:
1485: se.setContentEncoding(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
1486: post.setEntity(se);
1487: } //try
1488: catch (UnsupportedEncodingException e1) {
1489: e1.printStackTrace();
1490: }
1491:
1492: try{
1493: HttpResponse response = httpClient4.execute(post);
1494:
1495: //System.out.println("Status: " + response.getStatusLine().getStatusCode());
1496:
1497: if (response.getStatusLine().getStatusCode() == HttpStatus.SC_UNAUTHORIZED) {
1498: if (response.containsHeader("WWW-Authenticate")) {
1499: final Header challengeHeader = response.getHeaders("WWW-Authenticate")[0];
1500: DigestScheme ds = new DigestScheme();

```

```

main3.java
1501: ds.processChallenge(challengeHeader);
1502: final Header authHeader = ds.authenticate(
1503: new UsernamePasswordCredentials("user2", "ggsecret"),
1504: new BasicHttpRequest(HttpPost.METHOD_NAME, url));
1505:
1506: post.addHeader(authHeader);
1507:
1508: HttpClient httpClient5 = new DefaultHttpClient(cm, params);
1509:
1510: response = httpClient5.execute(post);
1511:
1512: /*
1513: System.out.println("Status: " + response.getStatusLine().getStatusCode());
1514:
1515: HttpResponse end = null;
1516: String endResult = null;
1517: end = response;
1518:
1519: BasicResponseHandler myHandler = new BasicResponseHandler();
1520:
1521: try {
1522: endResult = myHandler.handleResponse(end);
1523: System.out.println("Result: " + endResult);
1524: } catch (HttpResponseException e) {
1525: e.printStackTrace();
1526: } catch (IOException e) {
1527: e.printStackTrace();
1528: }
1529: */
1530:
1531: httpClient5.getConnectionManager().shutdown();
1532: httpClient4.getConnectionManager().shutdown();
1533:
1534: if ( response.getStatusLine().getStatusCode() == HttpStatus.SC_UNSUPPORTED_MEDIA_TYPE) {
1535: System.out.println("Error: " + response.getStatusLine().getStatusCode());
1536: }
1537: }
1538: }
1539: }//try
1540:
1541: catch(Exception e) {
1542: System.out.println("Error " + e);
1543: }//catch
1544: //Looper.loop();
1545: }//run
1546:];//thread
1547: t.start();
1548: }//testPerson

1549: }//class

```

```

settings.java
1: package mil.army.arl.hdpt;
2:
3: /*
4: import android.app.Activity;
5: import android.os.Bundle;
6: import android.widget.TextView;
7:
8: public class settings extends Activity {
9: public void onCreate(Bundle savedInstanceState) {
10: super.onCreate(savedInstanceState);
11:
12: TextView textview = new TextView(this);
13: textview.setText("Settings Tab");
14: setContentView(textview);
15: }
16: }
17: */
18:
19: import android.app.Activity;
20: import android.os.Bundle;
21: import android.view.View;
22: import android.view.View.OnClickListener;
23: import android.widget.Button;
24: import android.widget.CheckBox;
25: import android.widget.Toast;
26:
27: public class settings extends Activity implements OnClickListener
28: {
29:
30: Button b1;
31: CheckBox c1, c2, c3;
32: Toast msg;
33:
34: //default
35: public static String url_all = "http://10.1.200.82:8443/gg/";

```

```

36:
37: public void onCreate(Bundle icicle)
38: {
39:     super.onCreate(icicle);
40:     setContentView(R.layout.settings);
41:
42:     c1 = (CheckBox) findViewById(R.id.check1);
43:     c1.setChecked(false);
44:
45:     c2 = (CheckBox) findViewById(R.id.check2);
46:     c2.setChecked(false);
47:
48:     c3 = (CheckBox) findViewById(R.id.check3);
49:     c3.setChecked(true);
50:
51:     b1 = (Button) findViewById(R.id.button1);
52:     b1.setOnClickListener(this);
53: }
54:
55:
56: public void onClick(View arg0)
57: {
58:     if (c1.isChecked() == true && c2.isChecked() == true && c3.isChecked() == true) ||
59:         (c1.isChecked() == false && c2.isChecked() == true && c3.isChecked() == true) ||
60:         (c1.isChecked() == true && c2.isChecked() == true && c3.isChecked() == false) ||
61:         (c1.isChecked() == true && c2.isChecked() == false && c3.isChecked() == true) )
62:     {
63:         msg = Toast.makeText(settings.this,
64: "Please Select Only One Server", Toast.LENGTH_SHORT);
65:         msg.show();
66:     }
67:
68:     if (c1.isChecked() == true && c2.isChecked() == false && c3.isChecked() == false)
69:     {
70:
71:         msg = Toast.makeText(settings.this,
72: "Zulu - APG", Toast.LENGTH_SHORT);
73:         url_all="https://zulu.idvxn.arl.army.mil:8443/gg/";
74:         msg.show();
75:     }
76:
settings.java
76:
77: if (c1.isChecked() == false && c2.isChecked() == true && c3.isChecked() == false)
78: {
79:     msg = Toast.makeText(settings.this,
80: "Zulu - OTM", Toast.LENGTH_SHORT);
81:     url_all="https://10.1.200.87:8443/gg/";
82:     msg.show();
83: }
84:
85: if (c1.isChecked() == false && c2.isChecked() == false && c3.isChecked() == true)
86: {
87:     msg = Toast.makeText(settings.this,
88: "Command Web - OTM", Toast.LENGTH_SHORT);
89:     url_all="http://10.1.200.82:8443/gg/";
90:     msg.show();
91: }
92:
93: if (c1.isChecked() == false && c2.isChecked() == false && c3.isChecked() == false)
94: {
95:     msg = Toast.makeText(settings.this,
96: "No Server Selected", Toast.LENGTH_SHORT);
97:     msg.show();
98: }
99:
100: }
101: }

102: }611821

```

form.xml

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3: android:id="@+id/ScrollView01"
4: android:layout_width="fill_parent"
5: android:layout_height="wrap_content"
6: android:scrollbars="vertical" >
7:
8: <LinearLayout
9: android:layout_width="fill_parent"
10: android:layout_height="wrap_content"
11: android:orientation="vertical" >
12:
13: <!--
14: <TextView android:id="@+id/TextViewTitle" android:layout_width="wrap_content"
15: android:layout_height="wrap_content" android:text="@string/title" android:textSize="10pt"></TextView>
16: -->
17:
18: <TextView
19: android:id="@+id/label"
20: android:layout_width="fill_parent"
21: android:layout_height="wrap_content"
22: android:text="@string/name" />
23:
24: <EditText
25: android:id="@+id/EditTextName"
26: android:layout_width="320dip"
27: android:layout_height="wrap_content"
28: android:hint="@string/reportname" >
29: </EditText>
30:
31: <TextView
32: android:id="@+id/label1.1"
33: android:layout_width="fill_parent"
34: android:layout_height="wrap_content"
35: android:text="@string/nationality" />
36:
37: <Spinner
38: android:id="@+id/nationality_spinner"
39: android:layout_width="320dip"
40: android:layout_height="wrap_content"
41: android:hint="@string/reportnationality"
42: android:lines="1" >
43: </Spinner>
44:
45: <TextView
46: android:id="@+id/label1.15"
47: android:layout_width="fill_parent"
48: android:layout_height="wrap_content"
49: android:text="@string/tribe" />
50:
51: <Spinner
52: android:id="@+id/tribe_spinner"
53: android:layout_width="320dip"
54: android:layout_height="wrap_content"
55: android:hint="@string/reporttribe"
56: android:lines="1" >
57: </Spinner>
58:
59: <TextView
60: android:id="@+id/label1.2"
61: android:layout_width="fill_parent"
62: android:layout_height="wrap_content"
63: android:text="@string/gender" />
64:
65: <Spinner
66: android:id="@+id/gender_spinner"
67: android:layout_width="320dip"
68: android:layout_height="wrap_content"
69: android:hint="@string/reportgender"
70: android:lines="1" >
71: </Spinner>
72:
73: <TextView
74: android:id="@+id/label1.3"
75: android:layout_width="fill_parent"
```

```
form.xml
76: android:layout_height="wrap_content"
77: android:text="@string/marital" />
78:
79: <Spinner
80: android:id="@+id/marital_spinner"
81: android:layout_width="320dip"
82: android:layout_height="wrap_content"
83: android:hint="@string/reportmarital"
84: android:lines="1" >
85: </Spinner>
86:
87: <TextView
88: android:id="@+id/label1.4"
89: android:layout_width="fill_parent"
90: android:layout_height="wrap_content"
91: android:text="@string/birthplace" />
92:
93: <Spinner
94: android:id="@+id/birthplace_spinner"
95: android:layout_width="320dip"
96: android:layout_height="wrap_content"
97: android:hint="@string/reportbirthplace"
98: android:lines="1" >
99: </Spinner>
100:
101: <!--
102: <TextView
103: android:id="@+id/label1.5"
104: android:layout_width="fill_parent"
105: android:layout_height="wrap_content"
106: android:text="@string/height" />
107:
108: <Spinner
109: android:id="@+id/height_spinner"
110: android:layout_width="320dip"
111: android:layout_height="wrap_content"
112: android:hint="@string/reportheight"
113: android:lines="1" >
114: </Spinner>
115: -->
116:
117:
118: <!--
119: <TextView
120: android:id="@+id/label1.6"
121: android:layout_width="fill_parent"
122: android:layout_height="wrap_content"
123: android:text="@string/weight" />
124:
125: <Spinner
126: android:id="@+id/weight_spinner"
127: android:layout_width="320dip"
128: android:layout_height="wrap_content"
129: android:hint="@string/reportweight"
130: android:lines="1" >
131: </Spinner>
132: -->
133:
134: <TextView
135: android:id="@+id/label1.7"
136: android:layout_width="fill_parent"
137: android:layout_height="wrap_content"
138: android:text="@string/equipment" />
139:
140: <Spinner
141: android:id="@+id/equipment_spinner"
142: android:layout_width="320dip"
143: android:layout_height="wrap_content"
144: android:hint="@string/reportequipment"
145: android:lines="1" >
146: </Spinner>
147:
148: <TextView
149: android:id="@+id/label1.8"

150: android:layout_width="fill_parent"
```

form.xml

```
151: android:layout_height="wrap_content"
152: android:text="@string/vehicle" />
153:
154: <Spinner
155: android:id="@+id/vehicle_spinner"
156: android:layout_width="320dip"
157: android:layout_height="wrap_content"
158: android:hint="@string/reportvehicle"
159: android:lines="1" >
160: </Spinner>
161:
162: <TextView
163: android:id="@+id/label1.9"
164: android:layout_width="fill_parent"
165: android:layout_height="wrap_content"
166: android:text="@string/criminalrecord" />
167:
168: <Spinner
169: android:id="@+id/criminalrecord_spinner"
170: android:layout_width="320dip"
171: android:layout_height="wrap_content"
172: android:hint="@string/reportcriminalrecord"
173: android:lines="1" >
174: </Spinner>
175:
176: <TextView
177: android:id="@+id/label2.0"
178: android:layout_width="fill_parent"
179: android:layout_height="wrap_content"
180: android:text="@string/education" />
181:
182: <Spinner
183: android:id="@+id/education_spinner"
184: android:layout_width="320dip"
185: android:layout_height="wrap_content"
186: android:hint="@string/reporteducation"
187: android:lines="1" >
188: </Spinner>
189:
190: <TextView
191: android:id="@+id/label2.1"
192: android:layout_width="fill_parent"
193: android:layout_height="wrap_content"
194: android:text="@string/employment" />
195:
196: <Spinner
197: android:id="@+id/employment_spinner"
198: android:layout_width="320dip"
199: android:layout_height="wrap_content"
200: android:hint="@string/reportemployment"
201: android:lines="1" >
202: </Spinner>
203:
204: <TextView
205: android:id="@+id/label2.2"
206: android:layout_width="fill_parent"
207: android:layout_height="wrap_content"
208: android:text="@string/religion" />
209:
210: <Spinner
211: android:id="@+id/religion_spinner"
212: android:layout_width="320dip"
213: android:layout_height="wrap_content"
214: android:hint="@string/reportreligion"
215: android:lines="1" >
216: </Spinner>
217:
218: <TextView
219: android:id="@+id/label2.3"
220: android:layout_width="fill_parent"
221: android:layout_height="wrap_content"
222: android:text="@string/skill" />
223:
224: <Spinner
225: android:id="@+id/skill_spinner"
```

form.xml

```
226: android:layout_width="320dip"
227: android:layout_height="wrap_content"
228: android:hint="@string/reportskill"
229: android:lines="1" >
230: </Spinner>
231:
232: <TextView
233: android:id="@+id/label2.4"
234: android:layout_width="fill_parent"
```

```

235: android:layout_height="wrap_content"
236: android:text="@string/address" />
237:
238: <Spinner
239: android:id="@+id/address_spinner"
240: android:layout_width="320dp"
241: android:layout_height="wrap_content"
242: android:hint="@string/reportaddress"
243: android:lines="1" >
244: </Spinner>
245:
246: <TextView
247: android:id="@+id/label3.0"
248: android:layout_width="fill_parent"
249: android:layout_height="wrap_content"
250: android:text="@string/date" />
251:
252: <LinearLayout
253: android:id="@+id/button_bar"
254: android:layout_width="fill_parent"
255: android:layout_height="wrap_content"
256: android:orientation="horizontal" >
257:
258: <TextView
259: android:id="@+id/dateDisplay"
260: android:layout_width="fill_parent"
261: android:layout_height="wrap_content"
262: android:layout_weight="2"
263: android:textSize="10pt" />
264:
265: <Button
266: android:id="@+id/pickDate"
267: android:layout_width="fill_parent"
268: android:layout_height="wrap_content"
269: android:layout_weight="2"
270: android:text="@string/changedate" />
271: </LinearLayout>
272:
273: <Button
274: android:id="@+id/ButtonSendFeedback"
275: android:layout_width="fill_parent"
276: android:layout_height="wrap_content"
277: android:layout_marginTop="30dp"
278: android:text="@string/reportbutton" />
279: </LinearLayout>
280:

281: </ScrollView>
```

main.xml

```

1: <?xml version="1.0" encoding="utf-8"?>
2: <TabHost xmlns:android="http://schemas.android.com/apk/res/android"
3: android:id="@+id/tabhost"
4: android:layout_width="fill_parent"
5: android:layout_height="fill_parent" >
6:
7: <LinearLayout
8: android:layout_width="fill_parent"
9: android:layout_height="fill_parent"
10: android:orientation="vertical"
11: android:padding="5dp" >
12:
13: <TabWidget
14: android:id="@+id/tabs"
15: android:layout_width="fill_parent"
16: android:layout_height="wrap_content" />
17:
18: <FrameLayout
19: android:id="@+id/tabcontent"
20: android:layout_width="fill_parent"
21: android:layout_height="fill_parent"
22: android:padding="5dp" />
23: </LinearLayout>
24:

25: </TabHost>
```

```

modify.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3: android:id="@+id/ScrollView01"
4: android:layout_width="wrap_content"
5: android:layout_height="wrap_content"
6: android:scrollbars="vertical" >
7:
8: <LinearLayout
9: android:layout_width="fill_parent"
10: android:layout_height="wrap_content"
11: android:orientation="vertical" >
12:
13: <TextView
14: android:id="@+id/modifyperson_title"
15: android:layout_width="wrap_content"
16: android:layout_height="wrap_content"
17: android:text="@string/modifytitle"
18: android:textsize="8pt" >
19: </TextView>
20:
21: <Spinner
22: android:id="@+id/modifyperson_spinner"
23: android:layout_width="320dp"
24: android:layout_height="wrap_content"
25: android:hint="@string/modifyperson"
26: android:lines="1" >
27: </Spinner>
28:
29: <Button
30: android:id="@+id/modifyperson_button"
31: android:layout_width="fill_parent"
32: android:layout_height="wrap_content"
33: android:layout_marginTop="30dp"
34: android:onClick="modifyperson"
35: android:text="@string/modifybutton" >
36: </Button>
37: </LinearLayout>
38:

39: </ScrollView>

person_create.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <selector xmlns:android="http://schemas.android.com/apk/res/android">
3: <!-- When selected, use grey -->
4: <item android:drawable="@drawable/person_grey"
5: android:state_selected="true" />
6: <!-- When not selected, use white-->
7: <item android:drawable="@drawable/person_white" />

8: </selector>

person_modify.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <selector xmlns:android="http://schemas.android.com/apk/res/android">
3: <!-- When selected, use grey -->
4: <item android:drawable="@drawable/modify_grey"
5: android:state_selected="true" />
6: <!-- When not selected, use white-->
7: <item android:drawable="@drawable/modify_white" />

8: </selector>

settings.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3: android:id="@+id/LinearLayout01"
4: android:layout_width="fill_parent"
5: android:layout_height="fill_parent"
6: android:orientation="vertical" >
7:
8: <CheckBox
9: android:id="@+id/check1"
10: android:layout_width="300dp"
11: android:layout_height="wrap_content"
12: android:text="@string/server1" />
13:
14: <CheckBox
15: android:id="@+id/check2"
16: android:layout_width="300dp"
17: android:layout_height="wrap_content"
18: android:text="@string/server2" />
19:
20: <CheckBox
21: android:id="@+id/check3"

```

```

22: android:layout_width="300dp"
23: android:layout_height="wrap_content"
24: android:text="@string/server3" />
25:
26: <Button
27: android:id="@+id/button1"
28: android:layout_width="200dp"
29: android:layout_height="wrap_content"
30: android:text="@string/submitbutton" />
31:

32: </LinearLayout>

splash.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3: android:orientation="vertical"
4: android:layout_width="fill_parent"
5: android:layout_height="fill_parent">
6: <ImageView
7: android:layout_width="fill_parent"
8: android:layout_height="fill_parent"
9: android:scaleType="fitCenter"
10: android:src="@drawable/ar1"/>

11: </LinearLayout>

strings.xml
1: <?xml version="1.0" encoding="utf-8"?>
2: <resources>
3: <string name="app_name">HDPT Report</string>
4: <string name="name">Name:</string>
5: <string name="date">Date of Birth:</string>
6: <string name="nationality">Nationality:</string>
7: <string name="gender">Gender:</string>
8: <string name="marital">Marital Status:</string>
9: <string name="birthplace">Place of Birth:</string>
10: <!-- <string name="height">Height:</string> -->
11: <!-- <string name="weight">Weight:</string> -->
12: <string name="equipment">Equipment:</string>
13: <string name="vehicle">Vehicle:</string>
14: <string name="criminalrecord">Criminal Record:</string>
15: <string name="education">Education:</string>
16: <string name="employment">Employment:</string>
17: <string name="religion">Religion:</string>
18: <string name="skill">Skill:</string>
19: <string name="address">Address:</string>
20: <string name="tribe">Tribal Affiliation:</string>
21:
22: <string name="modifyperson">Suspect Name</string>
23: <string name="modifytitle">Suspect Name To Modify</string>
24: <string name="modifybutton">Submit</string>
25:
26: <string name="title">Enter details to send to HDPT</string>
27: <string name="reportname">Suspect Name</string>
28: <string name="reportnationality">Suspect Nationality</string>
29: <string name="reportgender">Suspect Gender</string>
30: <string name="reportmarital">Suspect Marital Status</string>
31: <string name="reportbirthplace">Suspect Place of Birth</string>
32: <!-- <string name="reportheight">Suspect Height</string> -->
33: <!-- <string name="reportweight">Suspect Weight</string> -->
34: <string name="reportequipment">Suspect Equipment</string>
35: <string name="reportvehicle">Suspect Vehicle</string>
36: <string name="reportcriminalrecord">Suspect Criminal Record</string>
37: <string name="reporteducation">Suspect Education</string>
38: <string name="reportemployment">Suspect Employment</string>
39: <string name="reportreligion">Suspect Religion</string>
40: <string name="reportskill">Suspect Skill</string>
41: <string name="reportaddress">Suspect Address</string>
42: <string name="reporttribe">Suspect Tribal Affiliation</string>
43:
44: <string name="changedate">Change Date</string>
45:
46: <string name="server1">Zulu - APG</string>
47: <string name="server2">Zulu - OTM</string>
48: <string name="server3">Command Web - OTM</string>
49: <string name="submitbutton">Submit</string>
50:
51: <string-array name="nationality_array">
52: <item>Choose Suspect Nationality</item>
53: <item>Afghan</item>
54: <item>Arab</item>
55: </string-array>
56:
57: <string-array name="gender_array">
58: <item>Choose Suspect Gender</item>
59: <item>Male</item>
60: <item>Female</item>

```

```

61: </string-array>
62:
63: <string-array name="marital_array">
64: <item>Choose Suspect Marital Status</item>
65: <item>Single</item>
66: <item>Married</item>
67: </string-array>
68:
69: <string-array name="birthplace_array">
70: <item>Choose Suspect Place of Birth</item>
71: <item>Born in Area</item>
72: <item>Born outside Area</item>
73: </string-array>
74:

75: <string-array name="equipment_array">
```

strings.xml

```

76: <item>Choose Suspect Equipment</item>
77: <item>Pistol</item>
78: <item>Knife</item>
79: <item>Gang Colors</item>
80: <item>Bomb</item>
81: <item>Video Camera</item>
82: <item>Cell Phone</item>
83: <item>Uniform</item>
84: <item>Briefcase</item>
85: </string-array>
86:
87: <string-array name="vehicle_array">
88: <item>Choose Suspect Vehicle</item>
89: <item>White Panel Truck</item>
90: <item>Blue Motorcycle</item>
91: <item>Silver Compact Car</item>
92: <item>Blue Minivan</item>
93: <item>Grey Sedan</item>
94: <item>Brown Pickup Truck</item>
95: <item>Black SUV</item>
96: <item>Burgundy Luxury Sedan</item>
97: </string-array>
98:
99: <string-array name="criminalrecord_array">
100: <item>Choose Suspect Criminal Record </item>
101: <item>Has One</item>
102: <item>Does Not Have One</item>
103: </string-array>
104:
105: <string-array name="education_array">
106: <item>Choose Suspect Education </item>
107: <item>High</item>
108: <item>Low</item>
109: </string-array>
110:
111: <string-array name="employment_array">
112: <item>Choose Suspect Employment </item>
113: <item>Not Employed</item>
114: <item>White Collar</item>
115: <item>Blue Collar</item>
116: </string-array>
117:
118: <string-array name="religion_array">
119: <item>Choose Suspect Religion </item>
120: <item>Mild Theology</item>
121: <item>Radical Theology</item>
122: </string-array>
123:
124: <string-array name="skill_array">
125: <item>Choose Suspect Skill</item>
126: <item>Photography</item>
127: <item>Writing</item>
128: <item>Electrical</item>
129: <item>Mechanical</item>
130: <item>Computer</item>
131: <item>Driving</item>
132: <item>Financial</item>
133: </string-array>
134:
135: <string-array name="address_array">
136: <item>Choose Suspect Address</item>
137: <item>Times Square Village</item>
138: <item>Viet Nam Village</item>
139: <item>Vertol Village</item>
140: <item>Hanover Village</item>
141: <item>Cook Corner Village</item>
142: <item>Gredge Town</item>
143: <item>Utes Village</item>
144: <item>Horizon Village</item>
145: </string-array>
```

```

146:
147: <string-array name="tribe_array">
148: <item>Choose Suspect Tribal Affiliation</item>
149: <item>Pashtu</item>

150: <item>Baloch</item>

strings.xml
151: <item>Hazara</item>
152: <item>Tajik</item>
153: </string-array>
154:
155: <!--
156: <string-array name="height_array">
157: <item>Choose Suspect Height</item>
158: <item>5</item>
159: <item>5.5</item>
160: <item>6</item>
161: <item>6.5</item>
162: </string-array>
163: -->
164:
165: <!--
166: <string-array name="weight_array">
167: <item>Choose Suspect Weight</item>
168: <item>100</item>
169: <item>125</item>
170: <item>150</item>
171: <item>175</item>
172: <item>200</item>
173: </string-array>
174: -->
175:
176: <string-array name="modifyperson_array">
177: <item>Choose Suspect Name</item>
178: <!--
179: <item>Test User</item>
180: <item>Bahij As'l Tawfeek</item>
181: <item>Jalal Anas Kader</item>
182: <item>Hashim Fouad Ahmad</item>
183: <item>Nazli Gauhar Ajam</item>
184: <item>Gabr Hussein Ahmed</item>
185: <item>Shakira Nashwa Abujamal</item>
186: <item>Rafiq Saif-al-Din Karim</item>
187: <item>Farouk Ghayth El-Ghazzawy</item>
188: <item>Karam Imen Boullos</item>
189: <item>Haroun Salih Abdullah</item>
190: <item>Najwa Nadia Saggaf</item>
191: <item>Mostafa Tufayl Karimi</item>
192: <item>Jinai Qadir El-Ghazzawy</item>
193: <item>Rana Lubna Samara</item>
194: <item>Salim Mus'l ad Hakim</item>
195: <item>Majid Rusul Abujamal</item>
196: <item>Harun Ziad Boullos</item>
197: <item>Miraj Rashid Karimi</item>
198: <item>Sulaiman Badr Muhammad</item>
199: <item>Nasir Baki Saab</item>
200: <item>Ziyad Guda Sultan</item>
201: <item>Mostafa Faroog Darzi</item>
202: <item>Wafi Murtada Hakim</item>
203: <item>Adam Abdur-Rashid Abdullah</item>
204: -->
205: <item>Abu Navid Sultan</item>
206: <item>Rasul Anass Zaman</item>
207: <item>Khalilah Qismat Amirmoez</item>
208: <item>Ikram I'timad Abdullah</item>
209: <item>Habib Ala Ahmed</item>
210: <item>Rasul Zayn Mohammed</item>
211: <item>Ridha Mahdi El-Mofty</item>
212: <item>Sami Mis'id El-Ghazzawy</item>
213: <item>Hussain Mansoor El-Hashem</item>
214: <item>Zaman Noor Hakim</item>
215: <item>Yusuf Mehmud Samara</item>
216: <item>Amirah Sani El-Amin</item>
217: <item>Aali Abu Bakr Karim</item>
218: <item>Saif-al-Din Jinan Hakim</item>
219: <item>Harun Shahzad El-Mofty</item>
220: <!--
221: <item>Shareef Navid Nejem</item>
222: <item>Qusay Hameed El-Amin</item>
223: <item>Tamid I'timad Ajam</item>
224: <item>Guda Jalal Ali</item>

225: -->

```

```
strings.xml
226: </string-array>
227:
228: <string name="reportbutton">Send Report</string>
229: <!-- <string name="feedbackmessagebody_format">ATTN: HDPT\nDATE: %4$s\n\nMESSAGE: %1$s\n\nNAME: %2$s\n(%3$s)</string> -->
230: </resources>
```

INTENTIONALLLY LEFT BLANK.

List of Symbols, Abbreviations, and Acronyms

ARL	U.S. Army Research Laboratory
CISD	Computational Information Sciences Directorate
DCGS-A	distributed common ground system
HDPT	heterogeneous data proximity tool
HVI	high-valued individuals
ISD	Information Sciences Division
JSON	JavaScript object notation
OWF	ozone widget framework
PFI	Potomac Fusion
REST	representational state transfer
TIFB	Tactical Information Fusion Branch
WAR	Web Application Archive

NO. OF
COPIES ORGANIZATION

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

1 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL CIO LL

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

1 ARMY G2
(PDF) D WALSH

1 MULTISOURCE INFORMATION
(PDF) FUSION
RSRCH PROFESSOR
(EMERITUS) J LLINAS

24 DIR USARL
(14 PDF RDRL CII C
10 HC) B BODT
E BOWMAN
F BRUNDICK
J DUMER
T HANRATTY
C HANSEN
E HEILMAN
S KASE
M MITTRICK (1 PDF, 10 HC)
A NEIDERER
K OGAARD
J RICHARDSON
H ROY
M THOMAS